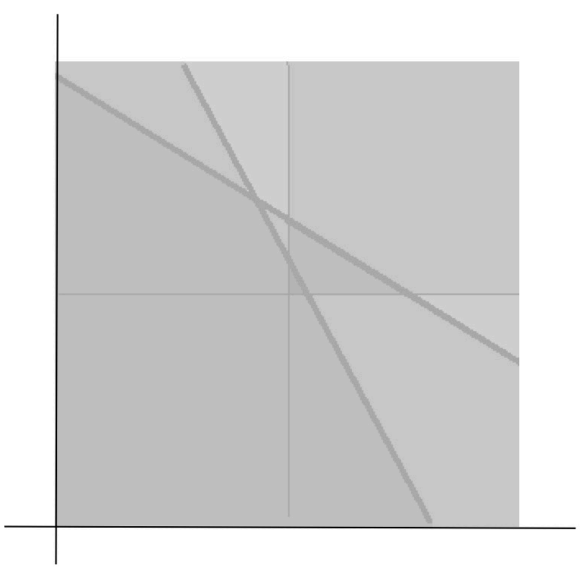
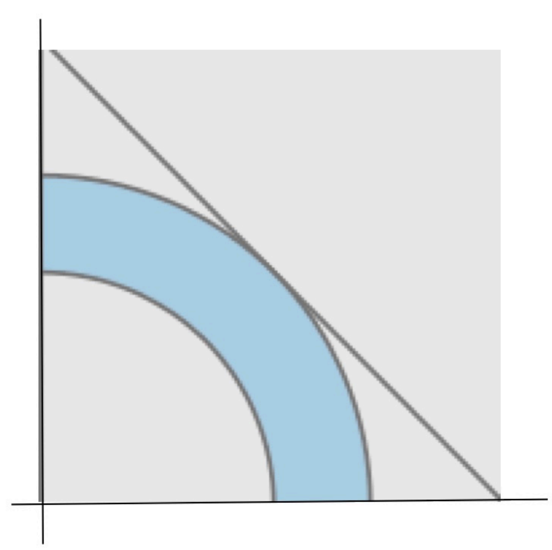
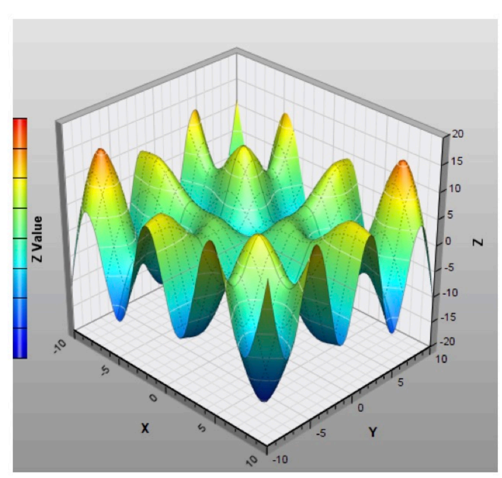
LINGO

Optimization Modeling Software for:
Linear
Nonlinear
and
Integer Programming

Solving Problems with LINGO







2019, Carlos Moya Mulero

1

CHAPTER

BLOCK 1

PREFACE

Of all, there were three things:

The certainty that we are always starting ...

The certainty that we need to continue ...

Rest assured that we will be interrupted before continuing ...

Therefore, we must:

Make of interruption, a path ...

Of the fall, a dance step ...

From fear, a ladder ...

From the dream, a bridge ...

Of the search, a meet. .

Fernando Pessoa

Operations management is at the heart of the great changes that are taking place in today's business environment.

Competitive pressures for higher quality, faster response time, superior service and full customization can only be met through more intelligently executed business operations.

The objective of this work is not to teach linear programming, but to present a collection of problems modeled in LINGO, in an auxiliary way in the development of new models or similar.

This type of modeling is designed for the use of large volumes of data, which can be made available in Database, Text Spreadsheets or Text Files, through OLE, ODBC and OLE functions.

The cases included in this work do not use external files, however, adjustments can be made to this by consulting the Lingo manual, available on Lindo Inc's website.

Thus, I hope that this content can be useful to the esteemed reader in the use of technology in their projects.

Author

BLOCK 2

LINGO

- Approach
- Key benefits of LINGO
- Overview
- Model Class
- Toolbar
- Menu Commands
- File type

C1-B2 LINGO APPROACH

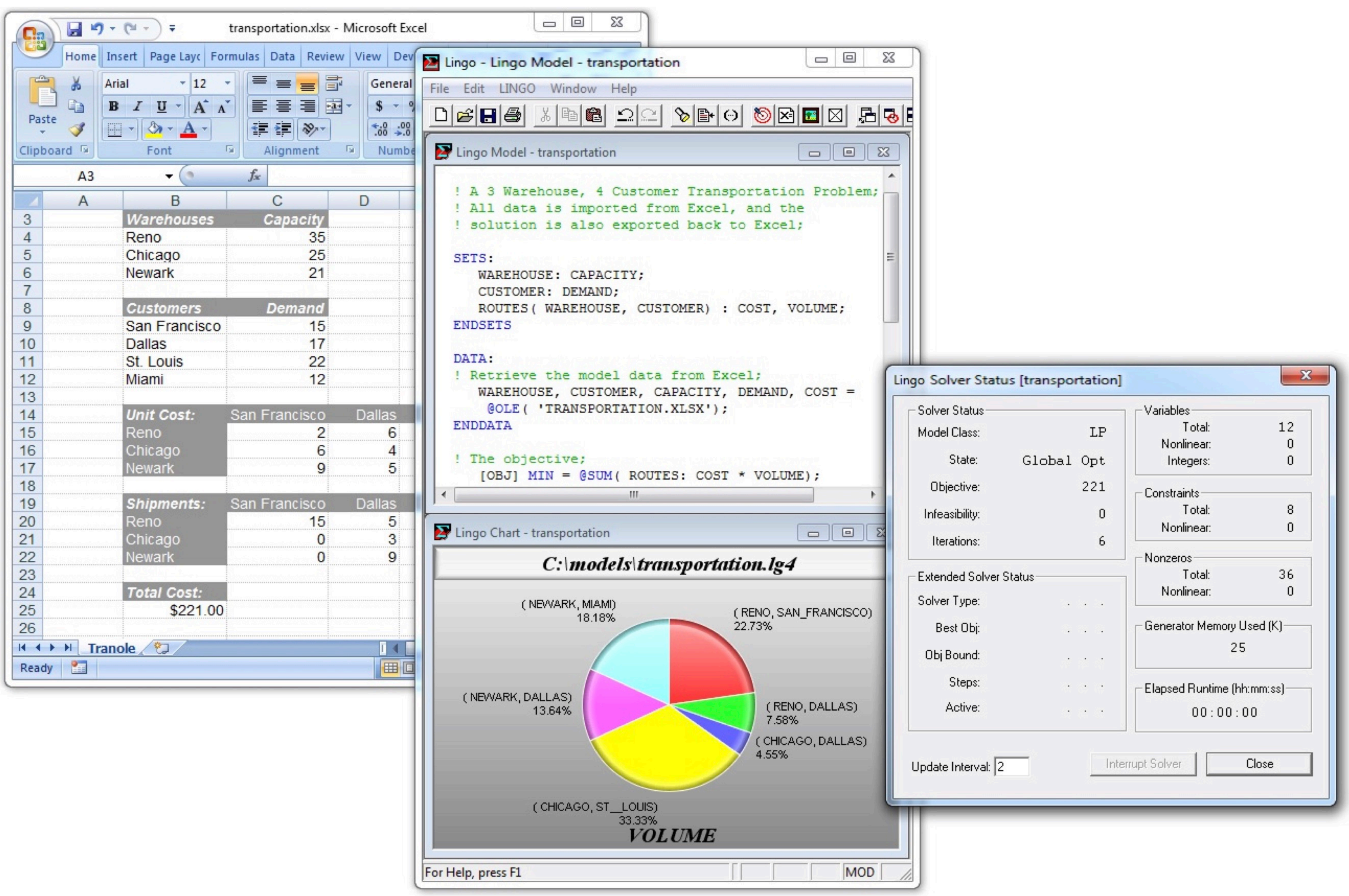
LINGO - Optimization Modeling Software for Linear, Nonlinear, and Integer Programming

LINGO combines a full featured modeling environment with a set of solvers for linear, integer, and nonlinear models. Beginning modelers will appreciate the easy, natural style of model expression and the comfortable Windows interface.

Power modelers will love the option to use the advanced modeling language, library of mathematical functions, and the ability to read data from spreadsheets, databases, an LINGO is a comprehensive tool designed to make building and solving Linear, Nonlinear (convex & nonconvex/Global), Quadratic, Quadratically Constrained, Second Order Cone, Semi-Definite, Stochastic, and Integer optimization models faster, easier and more efficient.

LINGO provides a completely integrated package that includes a powerful language for expressing optimization models, a full featured environment for building and editing problems, and a set of fast built-in solvers.

The recently released LINGO includes a number of significant enhancements and new features. Click here for more information on these new features.



EASY MODEL EXPRESSION

LINGO will help you cut your development time. It lets you formulate your linear, nonlinear and integer problems quickly in a highly readable form. LINGO's modeling language allows you to express models in a straightforward intuitive manner using summations and subscripted variables -- much like you would with pencil and paper. Models are easier to build, easier to understand, and, therefore, easier to maintain. LINGO can exploit multiple CPU cores for faster model generation.

CONVENIENT DATA OPTIONS

LINGO takes the time and hassle out of managing your data. It allows you to build models that pull information directly from databases and spreadsheets. Similarly, LINGO can output solution information right into a database or spreadsheet making it easier for you to generate reports in the application of your choice.

POWERFUL SOLVERS

LINGO is available with a comprehensive set of fast, built-in solvers for Linear, Nonlinear (convex & nonconvex/Global), Quadratic, Quadratically Constrained, Second Order Cone, Stochastic, and Integer optimization. You never have to specify or load a separate solver, because LINGO reads your formulation and automatically selects the appropriate one.

MODEL INTERACTIVELY OR CREATE TURN-KEY APPLICATIONS

You can build and solve models within LINGO, or you can call LINGO directly from an application you have written. For developing models interactively, LINGO provides a complete modeling environment to build, solve, and analyze your models. For building turn-key solutions, LINGO comes with callable DLL and OLE interfaces that can be called from user written applications. LINGO can also be called directly from an Excel macro or database application.

EXTENSIVE DOCUMENTATION AND HELP

LINGO provides all of the tools you will need to get up and running quickly. You get the LINGO User Manual (in printed form and available via the online help), which fully describes the commands and features of the program. Also included with super versions and larger is a copy of Optimization Modeling with LINGO, a comprehensive modeling text discussing all major BLOCKS of linear, integer and nonlinear optimization problems. LINGO also comes with dozens of real-world based examples for you to modify and expand.

DISTRIBUTORS U.S.	**DISTRIBUTORS BRAZIL**
U.S. and all countries not listed below:	PRODUTTARE Consultores Associados
LINDO Systems, Inc.	Rua Honorio Silveira Dias, 1550
1415 N. Dayton	Bairro Higienopolis
Chicago, IL 60642	90540-070 Porto Alegre RS Brasil
Contact: Mark Wiley	Contact: Flavio Pizzato
Tel: (312) 988-7422	Phone (fax) : +(55) 51 3024 5536
Fax: (312) 988-9065	+(55) 51 3022 8515

C1-B2 LINGO OVERVIEW

LINGO is a comprehensive tool designed to make building and solving mathematical optimization models easier and more efficient. LINGO provides a completely integrated package that includes a powerful language for expressing optimization models, a full-featured environment for building and editing problems, and a set of fast built-in solvers capable of efficiently solving most classes of optimization models. LINGO's primary features include:

Algebraic Modeling Language

LINGO supports a powerful, set-based modeling language that allows users to express math programming models efficiently and compactly. Multiple models may be solved iteratively using LINGO's internal scripting capabilities.

Convenient Data Options

LINGO takes the time and hassle out of managing your data. It allows you to build models that pull information directly from databases and spreadsheets. Similarly, LINGO can output solution information right into a database or spreadsheet making it easier for you to generate reports in the application of your choice. Complete separation of model and data enhance model maintenance and scalability.

Model Interactively or Create Turnkey Applications

You can build and solve models within LINGO, or you can call LINGO directly from an application you have written. For developing models interactively, LINGO provides a complete modeling environment to build, solve, and analyze your models. For building turn-key solutions, LINGO comes with callable DLL and OLE interfaces that can be called from user written applications. LINGO can also be called directly from an Excel macro or database application. LINGO currently includes programming examples for C/C++, FORTRAN, Java, C#.NET, VB.NET, ASP.NET, Visual Basic, Delphi, and Excel.

Extensive Documentation and Help

LINGO provides all of the tools you will need to get up and running quickly. You get the LINGO Users Manual (in printed form and available via the online Help), which fully describes the commands and features of the program. Also included with Super versions and larger is a copy of Optimization Modeling with LINGO, a comprehensive modeling text discussing all major classes of linear, integer and nonlinear optimization problems. LINGO also comes with dozens of real- world based examples for you to modify and expand.

Powerful Solvers and Tools

LINGO is available with a comprehensive set of fast, built-in solvers for linear, nonlinear (convex & nonconvex), quadratic, quadratically constrained, and integer optimization. You never have to specify or load a separate solver, because LINGO reads your formulation and automatically selects the appropriate one. A general description of the solvers and tools available in LINGO follows:

General Nonlinear Solver

LINGO provides both general nonlinear and nonlinear/integer capabilities. The nonlinear license option is required in order to use the nonlinear capabilities with LINDO API.

Global Solver

The global solver combines a series of range bounding (e.g., interval analysis and convex analysis) and range reduction techniques (e.g., linear programming and constraint propagation) within a branch-and-bound framework to find proven global solutions to non-convex nonlinear programs. Traditional nonlinear solvers can get stuck at suboptimal, local solutions. This is no longer the case when using the global solver.

Multistart Solver

The multistart solver intelligently generates a sequence of candidate starting points in the solution space of NLP and mixed integer NLPs. A traditional NLP solver is called with each starting point to find a local optimum. For non-convex NLP models, the quality of the best solution found by the multistart solver tends to be superior to that of a single solution from a traditional nonlinear solver. A user adjustable parameter controls the maximum number of multistarts to be performed.

Barrier Solver

The barrier solver is an alternative way for solving linear, quadratic and conic problems. LINGO's state-of-the-art implementation of the barrier method offers great speed advantages for large-scale, sparse models.

Simplex Solvers

LINGO offers two advanced implementations of the primal and dual simplex methods as the primary means for solving linear programming problems. Its flexible design allows the users to fine tune each method by altering several of the algorithmic parameters.

Mixed Integer Solver

The mixed integer solver's capabilities of LINGO extend to linear, quadratic, and general nonlinear integer models. It contains several advanced solution techniques such as cut generation, tree reordering to reduce tree growth dynamically, and advanced heuristic and resolve strategies.

Stochastic Solver

The stochastic programming solver supports decision making under uncertainty through multistage stochastic models with recourse. The user describes the uncertainty by identifying the distribution functions, either built-in or user-defined, describing each random variable. The stochastic solver will optimize the model to minimize the cost of the initial stage plus the expected cost of future recourse actions over the planning horizon. Advanced sampling modes are also available for approximating continuous distributions. LINGO's stochastic solver also supports chance-constrained models, where one or more sets of constraints are allowed to be violated according to a specified probability.

Model and Solution Analysis Tools

LINGO includes a comprehensive set of analysis tools for debugging infeasible linear, integer and nonlinear programs, using advanced techniques to isolate the source of infeasibilities to the smallest subset of the original constraints. It also has tools to perform sensitivity analysis to determine the sensitivity of the optimal basis to changes in certain data components (e.g. objective vector and right-hand-size values).

Quadratic Recognition Tools

The QP recognition tool is a useful algebraic pre-processor that automatically determines if an arbitrary NLP is actually a convex, quadratic model. QP models may then be passed to the faster quadratic solver, which is available as part of the barrier solver option. When the barrier solver option is combined with the global option, LINGO will automatically recognize conic models, in addition to convex quadratic models.

Linearization Tools

Linearization is a comprehensive reformulation tool that automatically converts many non-smooth functions and operators (e.g., max and absolute value) to a series of linear, mathematically equivalent expressions. Many non-smooth models may be entirely linearized. This allows the linear solver to quickly find a global solution to what would have otherwise been an intractable nonlinear problem.

C1-B2 LINGO VERSION

Maximum Problem Dimensions

Some versions of LINGO limit one or more of the following model properties: total variables, integer variables, nonlinear variables, global variables, and constraints. The total variable limit is on the total number of optimizable variables in your model (i.e., variables LINGO was unable to determine as being fixed at a particular value). The integer variable limit applies to the total number of optimizable variables restricted to being integers with either the @BIN or @GIN functions. The nonlinear variable limit applies to the number of optimizable variables that appear nonlinearly in the model's constraints. As an example, in the expression: X + Y, both X and Y appear linearly. However, in the expression:

X ^ 2 + Y, X appears nonlinearly while Y appears linearly. Thus, X would count against the nonlinear variable limit. In some cases, nonlinear variables are allowed only if you have purchased the nonlinear option for your LINGO software. The global variable limit applies to the total number of nonlinear variables when using the global solver. The constraint limit refers to the number of formulas in the model that contain one or more optimizable variables. Keep in mind that a single @FOR function may generate many constraints.

The maximum sized problem your LINGO can handle depends on the version you have. The current limits for the various versions are:

Version	Total Variables	Integer Variables	Nonlinear Variables	Global Variables	Constraints
Demo/Web	300	30	30	5	150
Solver Suite	500	50	50	5	250
Super	2,000	200	200	10	1,000
Hyper	8,000	800	800	20	4,000
Industrial	32,000	3,200	3,200	50	16,000
Extended	*Unlimited*	*Unlimited*	*Unlimited*	*Unlimited*	*Unlimited*

You can also determine the limits of your version by selecting the About LINGO command from the Help menu in Windows, or by typing HELP at the command-line prompt on other platforms. If you determine you need a larger version of LINGO, upgrades are available from LINDO Systems. Please feel free to contact us for pricing and availability.

C1-B2 MODEL CLASS

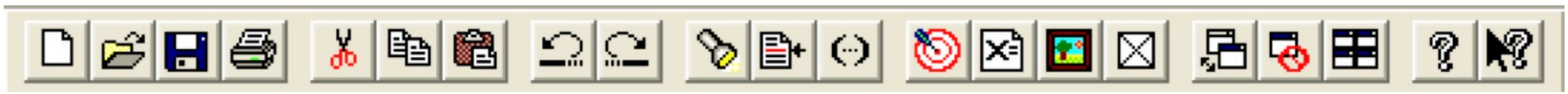
EXTENDED SOLVER STATUS BOX				
SOLVER TYPE:	BNP SOLVER	BRANCH-AND-BOUND	GLOBAL	MULTISTART
BEST OBJ:	The objective value of the best solution found so far.			
OBJ BOUND:	The theoretical bound on the objective.			
STEPS:	The number of steps taken by the extended solver.			
ACTIVE:	The number of active subproblems remaining to be analyzed.			

ABBREV	MODEL CLASS	DESCRIPTION MODEL CLASS FIELD
LP	LINEAR PROGRAM	All expressions are linear and the model contains no integer restrictions on the variables.
QP	QUADRATIC PROGRAM	All expressions are linear or quadratic and there are no integer restrictions.
CONE	CONIC PROGRAM	The model is a conic (second-order cone) program and all variables are continuous.
NLP	NONLINEAR PROGRAM	At least one of the relationships in the model is nonlinear with respect to the variables.
MILP	MIXED INTEGER LINEAR PROGRAM	All expressions are linear, and a subset of the variables is restricted to integer values.
MIQP	MIXED INTEGER QUADRATIC PROGRAM	All expressions are either linear or quadratic and a subset of the variables has integer restrictions.
MICONE	MIXED INTEGER CONIC PROGRAM	The model is a conic (second-order cone) program, and a subset of the variables is restricted to integer values.
MINLP	INTEGER NONLINEAR PROGRAM	At least one of the expressions in the model is nonlinear, and a subset of the variables has integer restrictions. In general, this class of model will be very difficult to solve for all but the smallest cases.
PILP	PURE INTEGER LINEAR PROGRAM	All expressions are linear, and all variables are restricted to integer values.
PIQP	PURE INTEGER QUADRATIC PROGRAM	All expressions are linear or quadratic and all variables are restricted to integer values.
PICONE	PURE INTEGER CONIC PROGRAM (Second-Order Cone)	The model is a conic (second-order cone) program, and all the variables are restricted to integer values.
PINLP	PURE INTEGER NONLINEAR PROGRAM	At least one of the expressions in the model is nonlinear, and all variables have integer restrictions. In general, this class of model will be very difficult to solve for all but the smallest cases.

C1-B2 LINGO TOOL BAR

The WindowsToolbar

LINGO's toolbar for its Windows version is illustrated in the following graphic:



Here is a list of the buttons and their equivalent commands:

File Menu:

File|New — *File|Open*

File|Save — *File|Print*

Edit Menu:

Edit|Undo — *Edit|Redo*

Edit|Cut — *Edit|Copy*

Edit|Paste — *Edit|Find*

Edit|Go To Line — *Edit|Match Parenthesis*

Solver Menu:

Solver|Solve — *Solver|Solution*

Solver|Options — *Solver|Picture*

Window Menu:

Window|Send To Back — *Window|Close All*

Window|Tile

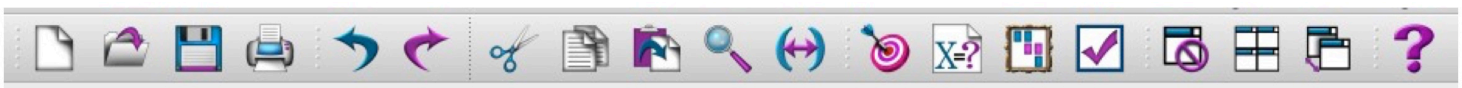
Help Menu:

Help|Topics — *Help|Pointer*

C1-B2 LINGO TOOL BAR

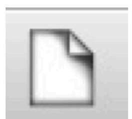
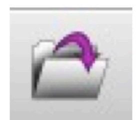
The Mac and LinuxToolbar

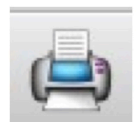
LINGO's toolbar for its Mac and Linux versions is illustrated in the following graphic:



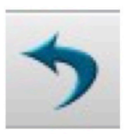
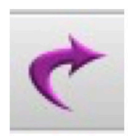
Here is a list of the buttons and their equivalent commands:

File Menu:

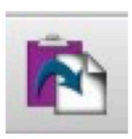
 *File|New*  *File|Open*

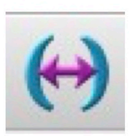
 *File|Save*  *File|Print*

Edit Menu:

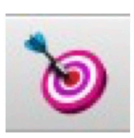
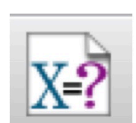
 *Edit|Undo*  *Edit|Redo*

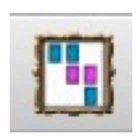
 *Edit|Cut*  *Edit|Copy*

 *Edit|Paste*  *Edit|Find*

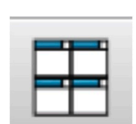
 *Edit|Match Parenthesis*

Solver Menu:

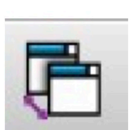
 *Solver|Solve*  *Solver|Solution*

 *Solver|Options*  *Solver|Picture*

Window Menu:

 *Window|Close All*  *Window|Tile*

Help Menu:

 *Window|Previous*  *Help|Topics*

File Menu Commands

Command				***Description***
New	✓	✓	✓	Opens a new model window.
Open	✓	✓	✓	Opens an existing model previously saved to disk.
Save	✓	✓	✓	Saves the contents of the current window to disk.
Save As	✓	✓	✓	Saves the contents of the current window to a new name.
Close	✓	✓	✓	Closes the current window.
Print	✓	✓	✓	Prints the contents of the current window.
Print Setup	✓	✓	✓	Configures your printer.
Print Preview	✓	✓	✓	Displays the contents of the current window as it would appear if printed.
Log Output	✓			Opens a log file for logging output to the command window.
Take Commands	✓			Runs a command script contained in a file.
Export File	✓	✓	✓	Exports a model in MPS or MPI file format.
License	✓	✓	✓	Prompts you for a new license password to upgrade your system.
Database User Info	✓			Prompts you for a user id and password for database access via the *@ODBC()* function.
Exit	✓	✓	✓	Exits LINGO.

Edit Menu Commands

Command				***Description***
Undo	✓	✓	✓	Undoes the last change.
Redo	✓	✓	✓	Redoes the last undo command.
Cut	✓	✓	✓	Cuts the current selection from the document.
Copy	✓	✓	✓	Copies the current selection to the clipboard.
Paste	✓	✓	✓	Pastes the contents of the clipboard into the document.
Paste Special	✓			Pastes the contents of the clipboard into the document, allowing choice as to how the object is pasted.
Select All	✓	✓	✓	Selects the entire contents of the current window.
Find	✓	✓	✓	Searches the document for the occurrence of a specified text string.
Find Next	✓	✓	✓	Repeats the find operation for the last string specified.
Replace	✓	✓	✓	Replaces a specified text string with a new string.
Go To Line	✓			Moves the cursor to a specified line number.
Match Parenthesis	✓	✓	✓	Finds the parenthesis that closes a selected parenthesis.
Paste Function	✓	✓	✓	Pastes a template of a selected LINGO function.
Select Font	✓	✓	✓	Specifies a font for a selected block of text.
Insert New Object	✓			Embeds an OLE object into the document.
Links	✓			Controls the links to external objects in your document.
Object Properties	✓			Specifies the properties of a selected, embedded object.

Solver Menu Commands

Command				***Description***
Solve	✓	✓	✓	Solves the model in the current window.
Solution	✓	✓	✓	Generates a solution report window for the current model.
Range	✓	✓	✓	Generates a range analysis report for the current window.
Options	✓	✓	✓	Sets system options.
Generate	✓	✓	✓	Generates the algebraic representation for the current model.
Picture	✓	✓	✓	Displays a graphical picture of the models matrix.
Debug	✓	✓	✓	Tracks down formulation errors in infeasible and unbounded linear programs.
Model Statistics	✓	✓	✓	Displays a brief report regarding the technical details of a model.
Look	✓			Generates a formulation report for the current window.

Window Menu Commands

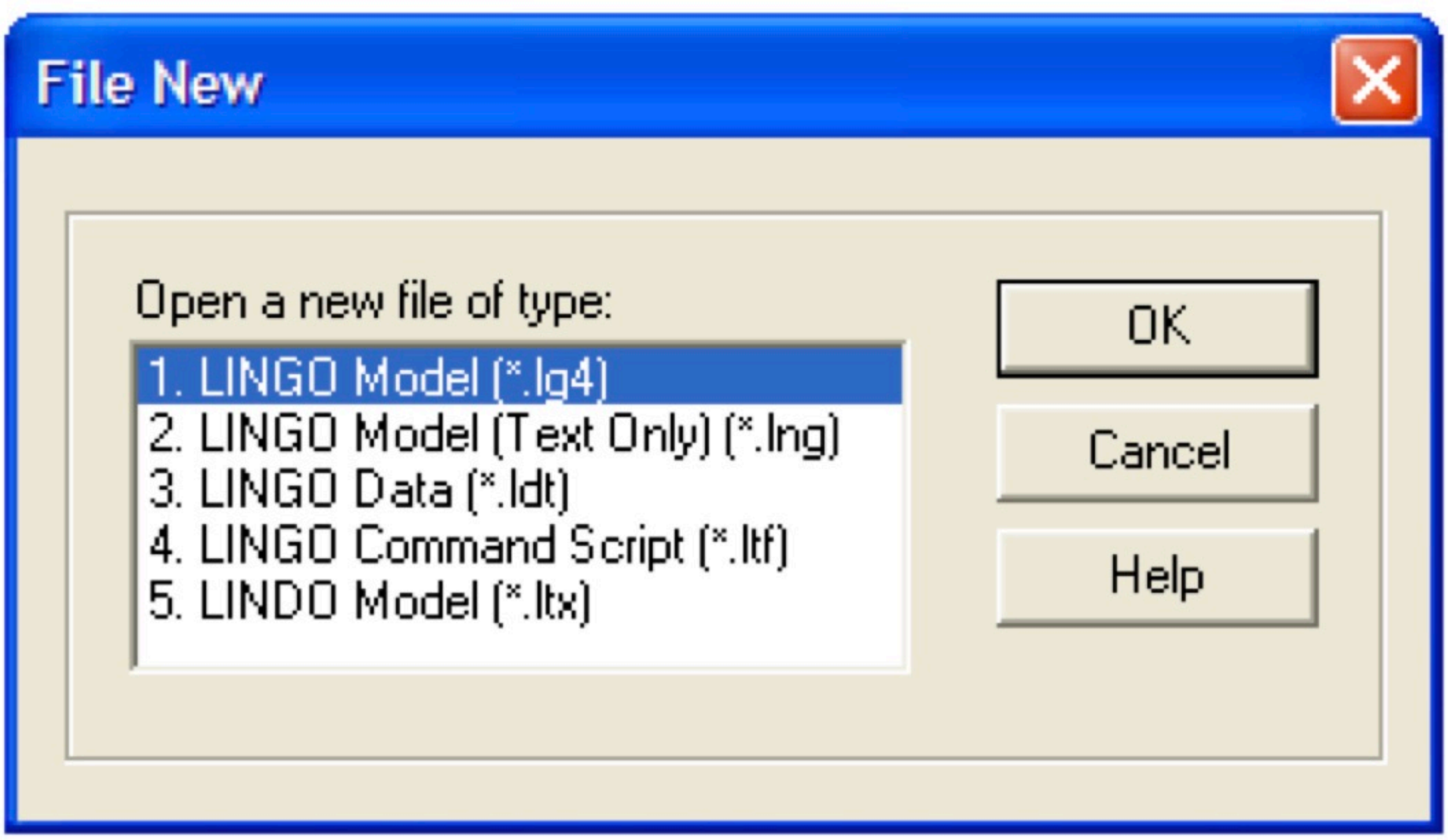
Command				***Description***
Command Window	✓	✓	✓	Opens a command window for command-line operation of LINGO.
Status Window	✓	✓	✓	Opens the solver's status window.
Close All	✓	✓	✓	Closes all open windows.
Tile	✓	✓	✓	Arranges all open windows into a tiled pattern.
Cascade	✓	✓	✓	Arranges all open windows into a cascading pattern.
Next		✓	✓	Brings the next window to the front.
Previous	✓	✓	✓	Brings the previous window to the front.
Arrange Icons	✓			Aligns all iconized windows at the bottom of the main

Help Menu Commands

Command				***Description***
Help Topics	✓	✓	✓	Accesses LINGO's online help facility.
Register	✓			Registers your version of LINGO online.
AutoUpdate	✓			Checks to see if an updated copy of LINGO is available for download on the LINDO Systems Web site
About LINGO	✓	✓	✓	Displays the version and size of your copy of LINGO, along Systems.with information on how to contact LINDO

File|New

The New command opens a new, blank window. When you select the New command, you will be presented with the following dialog box:



You may then select the type of file you want to create. The file must be one of the four types:

C1-B2 FILE EXTENSION

1. LINGO Model (*.lg4)

The LG4 format was established with release 4.0 of LINGO. LG4 is the primary file format used by LINGO to store models under Windows and is not used on other platforms. This format supports multiple fonts, custom formatting, and Object Linking and Embedding (OLE). LG4 files are saved to disk using a custom binary format. Therefore, these files can't be read directly into other applications or transported to platforms other than Windows. Use the LNG format (discussed next) to port a file to other applications or platforms.

2. LINGO Model (Text Only) (*.lng)

The LNG format is a portable format for storing your models. It is the standard file format used by LINGO on non-Windows platforms. LNG files are saved to disk as ASCII text and may be read into any application or word processor that supports text files. The LNG file format is supported by LINGO on all platforms, and LNG files can be ported from one platform to another. LNG files do not support multiple fonts, custom formatting, or OLE embedding of objects.

3. LINGO Data (*.ldt)

LDT files are data files that are typically imported into LINGO models using the @FILE function. @FILE can only read text files. Given this, all LDT files are stored as ASCII text. LDT files do not support multiple fonts, custom formatting, or OLE embedding.

4. LINGO Command Script (*.ltf)

LTF files are LINGO command scripts. These are ASCII text files containing a series of LINGO commands that can be executed with the File|Take Commands command. For more information on commands that can be used in a LINGO script, refer to Command-line Commands. LTF files do not support multiple fonts, custom formatting, or OLE.

5. LINDO Model (*.ltx)

LTX files are model files that use the LINDO syntax. Longtime LINDO users may prefer LINDO syntax over LINGO syntax. LINDO syntax is convenient for quickly entering small to medium-sized linear programs. As long as a file has an extension of .ltx, LINGO will assume that the model is written using LINDO syntax. Readers interested in the details of LINDO syntax may contact LINDO Systems to obtain a LINDO user's manual.

When you simply press either the New toolbar button or the F2 key, LINGO assumes you want a model file. Thus, LINGO does not display the file type dialog box and immediately opens a model file of type LG4. If you have used the Solver|Options command to change the default model file format from LG4 to LNG, LINGO will automatically open a model of type LNG when you press either the New button or the F2 key.

You may begin entering text directly into a new model window or paste in text from other applications using the Windows clipboard and the Edit|Paste command in LINGO.

BLOCK 3

Manual & Models

- Optimization Modeling
- User’s Manual
- Models

C1-B3 LINGO Optimization Modeling with LINGO by Linus Schrage

LINDO SYSTEMS INC.

Search ...

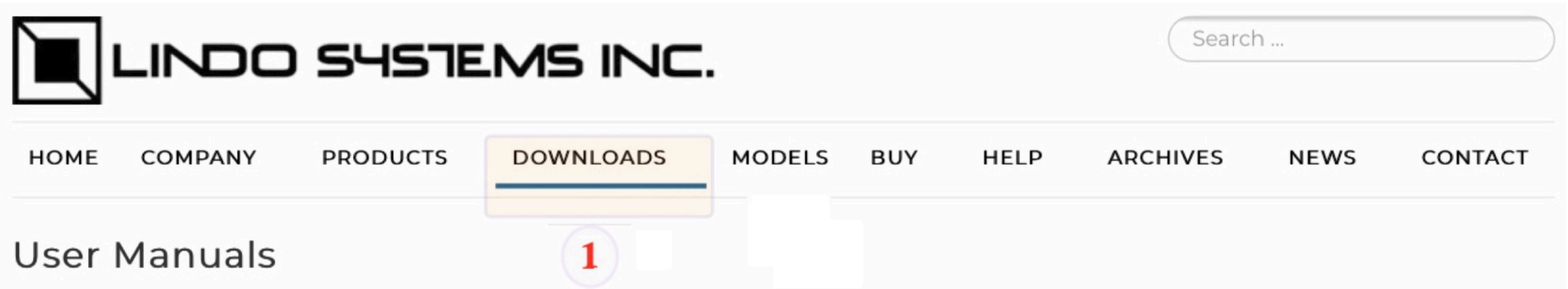
HOME COMPANY PRODUCTS DOWNLOADS MODELS BUY HELP ARCHIVES NEWS CONTACT

User Manuals 1

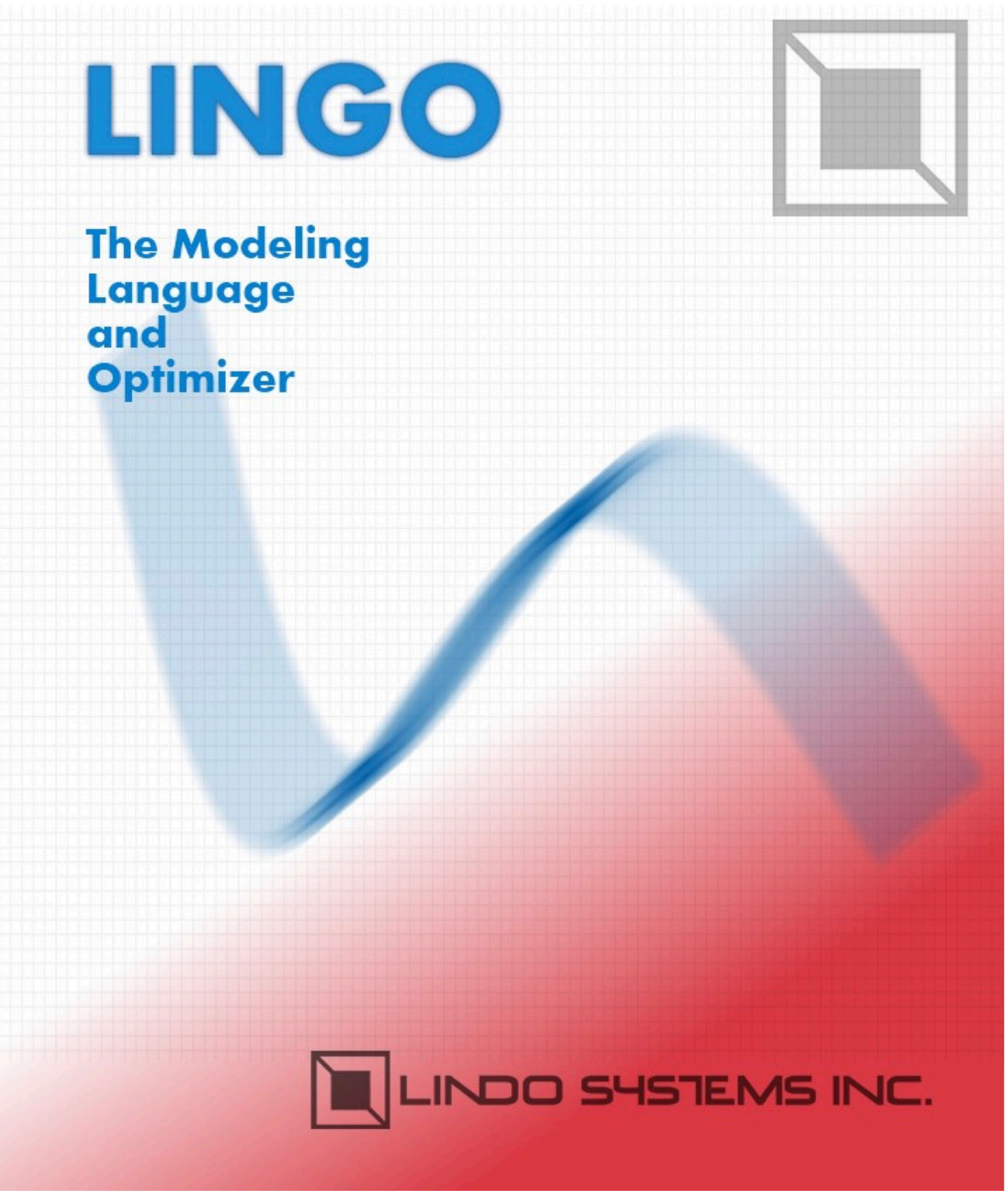
Click here to Download *Optimization Modeling with LINGO by Linus Schrage*

Optimization Modeling with LINGO by Linus Schrage

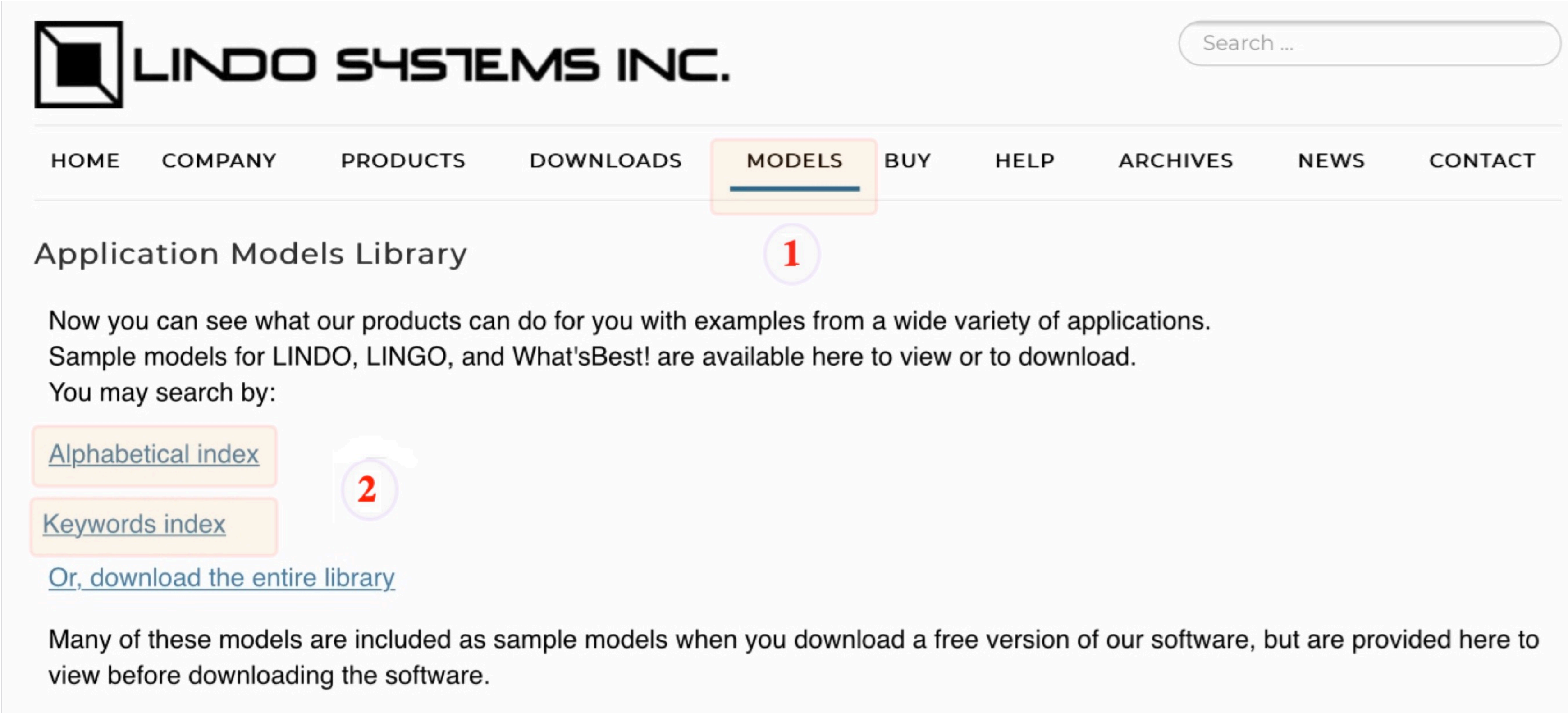
C1-B3 LINGO User's Manual

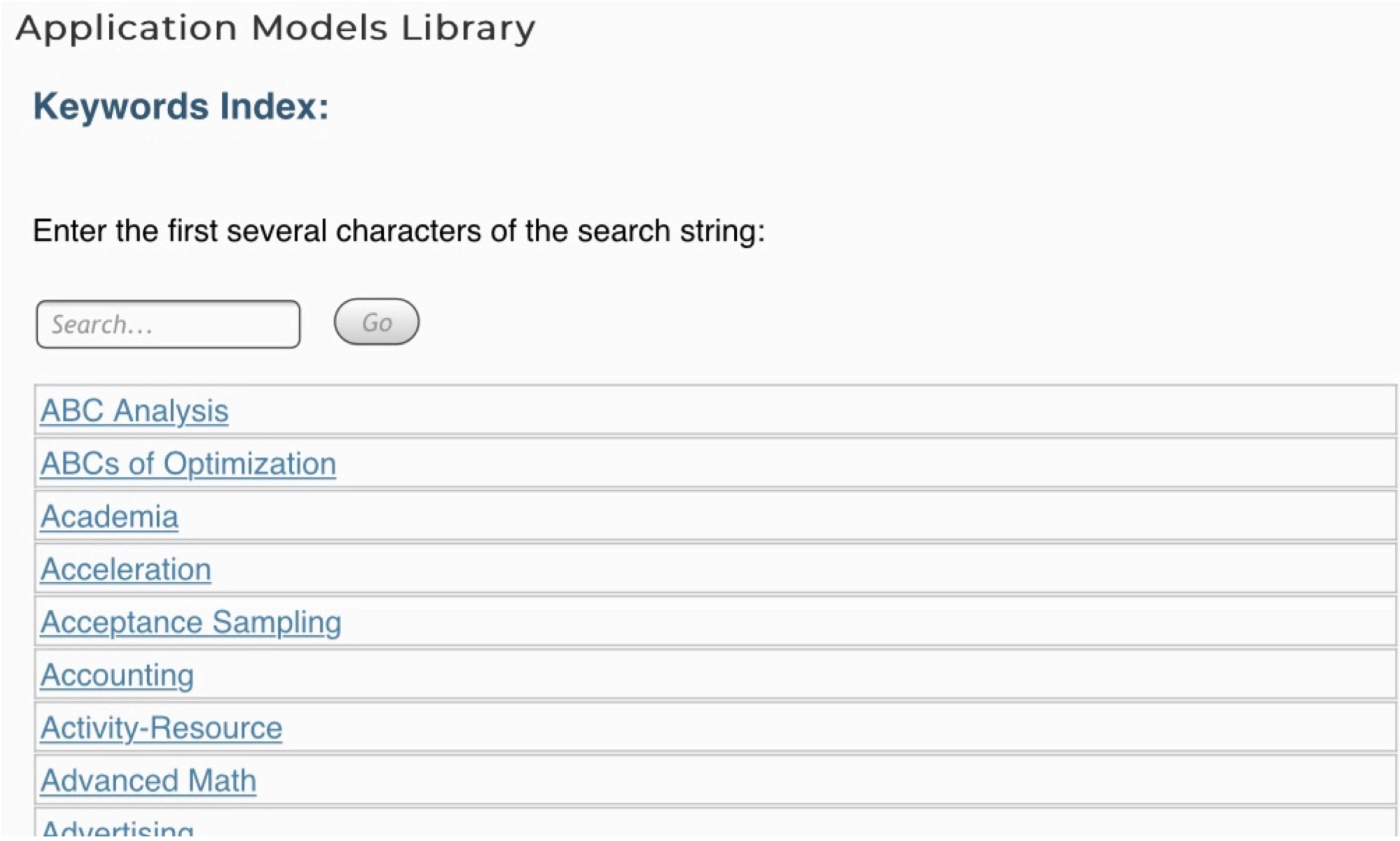


Download LINGO User's Manual (PDF)
LINGO is a comprehensive tool designed to help you build and solve linear, nonlinear, and integer optimization models quickly, easily, and efficiently. LINGO includes a powerful modeling language, a full-featured environment for building and editing problems, the ability to read and write to Excel and databases, and a set of fast built-in solvers.



C1-B3 LINGO Models





BLOCK 4 CONTENT

NUMBER OF CASES

Block	Name	Count	%
B1	Product Mix	27	20.61
B6	Transport	15	11.45
B5	Aviation	9	6.87
B2	Blend	9	6.87
B10	Schedule	8	6.11
B7	Agriculture	8	6.11
B12	Metallurgy	7	5.34
B14	Investment	6	4.58
B11	Cutting	6	4.58
B4	Diet	6	4.58
B9	Refinery	5	3.82
B3	Finance	4	3.05
B13	Fertilizer	4	3.05
B16	Classics	3	2.29
B15	Clinic	3	2.29
B18	Logistics	3	2.29
B8	Construction	3	2.29
B20	Assembly Line Balancing	2	1.53
B17	Dynamic	2	1.53
B19	Energy	1	0.76
Total		131	100.00

2

CHAPTER

BLOCK 1

BLOCKS: PRODUCT MIX

What should be the composition of products to be manufactured by a company in order to achieve maximum profit, with the limitations or requirements of the buyer market and the production capacity of the factory being respected?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Engines
- Plant

Source:

- Book 2
- Page 231

GOAL

An engine factory produces three models in its three factories: model 1100 cc, 1400 cc and 1800 cc. A union conflict foresees, in the short term, a prolonged strike at the factory 1.

To address this situation, the board decided to prepare a special production and sales plan for the next period, assuming there will be no production at factory 1 during the strike.

During this same period, the capacity of the Plant 2 is:

- 3000 engines of 1100 cc, if only this model were manufactured;
- 8000 engines of 1400 cc, if only this model were manufactured;
- 2000 engines of 1800 cc, if only this model were manufactured;
- Or any appropriate combination of these models.

During this same period, the capacity of Plant 3 is:

- 4000 engines of 1100 cc, if only this model were manufactured;
- 8000 engines of 1400 cc, if only this model were manufactured;
- Or any appropriate combination of these two models.
- The 1800 cc model is not produced in this factory

The company has entered into commitments that require it to supply at least 1000 engines of 1800 cc for export.

On the other hand, due to the decline in the domestic demand for 1100 cc engines and 1800 cc, the marketing department estimates 1000 and 1500 units of sales of these two models respectively.

Since the 1400 cc model is currently highly commercially successful, no limitations on its commercialization are envisaged. Determine the production plan that maximizes profit.

Motor Production		Models	Plant 2	Plant 3	Demand
Capacity	un	M1100	3000	4000	1000
	un	M1400	3000	8000	8000
	un	M1800	2000	0	1500
		Models	Cost		Price
Cost/Price	$	M1100	875.00	900.00	1,150.00
	$	M1400	1,260.00	1,100.00	1,450.00
	$	M1800	1,450.00	0.00	1,800.00

```
MODEL:
SETS:
MOTORS:DEMAND, PRICE ;
FACTORY:;
ROUTES( MOTORS, FACTORY): COST, VOLUME, CAPACITY;
ENDSETS
DATA:
! Products attributes;
MOTORS,                  DEMAND,     PRICE  =
M1100                    1000        1150
M1400                    8000        1450
M1800                    1500        1800;
! Resources attributes;
FACTORY      =
PLANT2
PLANT3;
! Engine cost            Plant2      Plant3;
COST         =           875         900          !  M1100;
                         1260        1100         !  M1400;
                         1450         0;          !  M1800;

! Manufacturing capacity Plant2      Plant3;
CAPACITY     =           3000        4000         !  M1100;
                         3000        8000         !  M1400;
                         2000        0;           !  M1800;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX = @SUM( ROUTES( I, J):PRICE(I) * VOLUME( I, J) - COST( I,J) * VOLUME( I,J));
! The demand constraints;
@FOR( MOTORS( I):
      [DEM] @SUM( FACTORY( J): VOLUME( I, J)) = DEMAND( I));
! The capacity constraints;
@FOR( ROUTES( I,J): VOLUME(I,J) <= CAPACITY(I,J));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  UNIT COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  CAPACITY (un):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  PRICE (un):", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1),"  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J)| VOLUME( I, J) #GT# 0: '  ',FACTORY( J),' produce:',
      @FORMAT(VOLUME( I, J),'%4g'),'un ', MOTORS( I), ' x Price:$',
      @FORMAT(PRICE(I),'%7.2f'), ' = Revenue:$',
      @FORMAT(PRICE(I) * VOLUME( I, J),'%11.2f'), @NEWLINE(1),46*' ', ' -    Cost:$',
      @FORMAT(COST( I,J) * VOLUME( I, J)),'%11.2f'), @NEWLINE(1),46*' ', ' =  Profit:$',
      @FORMAT((PRICE(I) * VOLUME( I, J)) - (COST( I,J) * VOLUME( I, J)),'%11.2f'),
@NEWLINE( 2));
@WRITE('  TOTAL:', 39*' ','= Revenue:$',
@FORMAT(@SUM(ROUTES(I,J): PRICE(I) * VOLUME( I, J)),'%11.2f'), @NEWLINE(1),46*' ', ' -    Cost:$',
@FORMAT(@SUM(ROUTES(I,J): COST( I,J) * VOLUME( I, J)),'%11.2f'), @NEWLINE(1),46*' ', ' =  Profit:$',
@FORMAT(@SUM(ROUTES(I,J): PRICE(I) * VOLUME( I, J) - (COST( I,J) * VOLUME( I, J))),'%11.2f'),@NEWLINE(2));
! Execute the graph;
 @CHARTSVBAR( 'Production capacity per plant', 'Plant','Unit', CAPACITY);
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

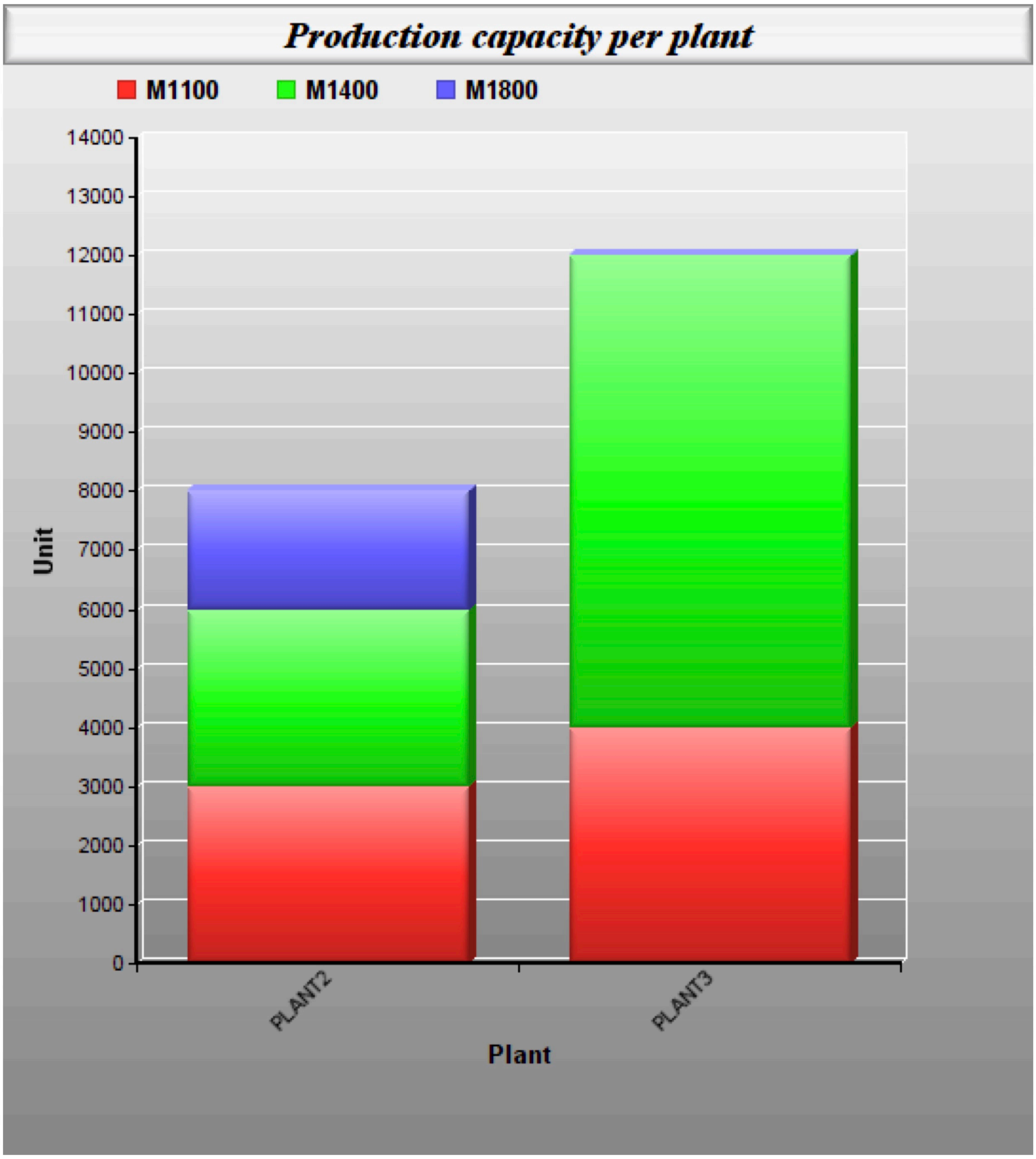
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
UNIT COST:
           PLANT2    PLANT3       DEMAND (un):
 M1100  875.0000  900.0000         M1100  1000.000
 M1400  1260.000  1100.000         M1400  8000.000
 M1800  1450.000  0.000000         M1800  1500.000

CAPACITY (un):
           PLANT2    PLANT3       PRICE (un):
 M1100  3000.000  4000.000         M1100  1150.000
 M1400  3000.000  8000.000         M1400  1450.000
 M1800  2000.000  0.000000         M1800  1800.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                 3600000.
Infeasibilities:                                                 0.00000

OPTIMAL PRODUCTION PROGRAM:
PLANT2 produce:1000un M1100 x Price:$1150.00 = Revenue:$ 1150000.00
                                             -    Cost:$  875000.00
                                             =  Profit:$  275000.00

PLANT3 produce:8000un M1400 x Price:$1450.00 = Revenue:$11600000.00
                                             -    Cost:$ 8800000.00
                                             =  Profit:$ 2800000.00

PLANT2 produce:1500un M1800 x Price:$1800.00 = Revenue:$ 2700000.00
                                             -    Cost:$ 2175000.00
                                             =  Profit:$  525000.00

TOTAL:                                       = Revenue:$15450000.00
                                             -    Cost:$11850000.00
                                             =  Profit:$ 3600000.00
```


Production capacity per plant
M1100
M1400
M1800
14000
13000
12000
11000
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Unit
PLANT2
PLANT3
Plant

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 224

GOAL

A shoemaker makes 6 shoes per hour, if he only makes shoes, and 5 belts per hour, if he only makes belts.

Resources / Products		Shoe	Belt	Available
Leather	un	2	1	6
Time Spent	min	10	12	60
Unit Profit	$	5.00	2.00	-

Formulate the model of the system of production of the cobbler, in order to obtain the maximum revenue per hour.

```
MODEL:
SETS:
 PRODUCT : PRICE, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,        AVAILABLE    =
 LEATHER          6
 TIMESPENT        60;

 ! Product attributes;
 PRODUCT,         PRICE        =
 SHOES            5
 BELTS            4;

! Required           SHOES        BELTS;
  USAGE    =         2            1        ! Leather (un);
                     10           12;      ! TimeSpent (min);
ENDDATA
SUBMODEL MAX2:
MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   REQUIRED (un, min):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (un, min):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR(PRODUCT(J): '  ', Product(J),', Produce:',
        @FORMAT(produce( J),'%2.0f'),'un x ', 'Unit Profit: $',
        @FORMAT(Price( J),'%4.2f'), ' = ', 'Total: $',
        @FORMAT(produce( J) * Price( J),'%5.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 REQUIRED (un, min):
                SHOES     BELTS
 LEATHER     2.000000  1.000000
 TIMESPENT   10.00000  12.00000

AVAILABLE (un, min):
 LEATHER     6.000000
 TIMESPENT   60.00000

PRICE:
 SHOES   5.000000
 BELTS   4.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                21.42857
Infeasibilities:                                0.000000

OPTIMAL PRODUCTION PROGRAM:
SHOES, Produce: 1un x Unit Profit: $5.00 = Total: $ 4.29
BELTS, Produce: 4un x Unit Profit: $4.00 = Total: $17.14
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 224

GOAL

One company manufactures 2 models of leather belts. The MOD1 model, of better quality, requires twice the manufacturing time compared to the MOD2 model (scale). If all the belts are of the model MOD2, at most 1000 units per day can be produced.

Resources / Products		Mod1	Mod2	Available
Raw Material (Buckle)	un	400	700	1,100
Production Scale	un	2	1	1,000
Unit Profit	$	4.00	3.00	-

What is the optimal production program that maximizes the company's total daily revenue?

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        AVAILABLE    =
 BUCKLE_A         400
 BUCKLE_B         700
 SCALE            1000;

 ! Products attributes;
 PRODUCT,         PROFIT       =
 MOD1             4
 MOD2             3;

! Required         MOD1         MOD2;
 USAGE =          1             0              ! BUCKLE_A;
                  0             1              ! BUCKLE_B;
                  2             1;             ! SCALE;
ENDDATA
SUBMODEL MAX3:
MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;

@SET('TERSEO',1);

! Post status windows, 1 Yes, 0 No;

@SET('STAWIN',0);

! Data block;

@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX3);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) | I #LE# 1:'  ', Product(J),', Produce:',
      @FORMAT(produce( J),'%4.0f'),'un using ',
      @FORMAT(RESOURCE(I),'-8s'), ' x ','Unit PROFIT:$',
      @FORMAT(PROFIT( J),'%4.2f'), ' = ','Total:$ ',
      @FORMAT(produce( J) * PROFIT( J),'%7.2f'),
@NEWLINE( 1));
! Execute the graph;
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE:
                 MOD1      MOD2
 BUCKLE_A   1.000000  0.000000
 BUCKLE_B   0.000000  1.000000
 SCALE      2.000000  1.000000

AVAILABLE:
 BUCKLE_A   400.0000
 BUCKLE_B   700.0000
 SCALE      1000.000

PRICE:
 MOD1       4.000000
 MOD2       3.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                    2700.000
Infeasibilities:                                    0.000000

OPTIMAL PRODUCTION PROGRAM:
MOD1, Produce: 150un using BUCKLE_A x Unit PROFIT:$4.00 = Total:$  600.00
MOD2, Produce: 700un using BUCKLE_A x Unit PROFIT:$3.00 = Total:$ 2100.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

Source:

- Book 2
- Page 225

GOAL

A company, after a process of rationalization of production, has available with 3 productive resources, R1, R2 and R3, that can be used to produce two products P1 and P2

Resources / Products			P1	P2	Available
Assembler Line	R1	un	2	4	100
	R2	un	3	2	90
	R3	un	5	3	120
Unit Profit		$	120.00	150.00	-

So, what is the monthly output of P1 and P2 that brings the highest revenue to the company?

```
MODEL:
SETS:
 PRODUCT : PRICE, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes ;
 RESOURCE,        AVAILABLE    =
 R1               100
 R2               90
 R3               120;
! Products attributes;
 PRODUCT,         PRICE        =
 P1               120
 P2               150;

! Required        P1           P2;
 USAGE =          2            4       ! R1;
                  3            2       ! R2;
                  5            3;      ! R3;
ENDDATA
SUBMODEL MAX4:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r);
      @GIN( PRODUCE ); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   PRODUCTIVE RESOURCES BY UNIT):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (un):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX4);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( product( J):'  ',
      @FORMAT(Product(J), '-2s'),':  ',
      @FORMAT(produce( J),'%2.0f'),'un  x  ', 'Unit Profit: $',
      @FORMAT(Price( J),'%4.2f'), '  =  ', 'Total: $',
      @FORMAT(produce( J) * Price( J),'%4.2f'),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 PRODUCTIVE RESOURCES BY UNIT:
           P1        P2
 R1   2.000000  4.000000
 R2   3.000000  2.000000
 R3   5.000000  3.000000

AVAILABLE (un):
 R1   100.0000
 R2   90.00000
 R3   120.0000

PRICE:
 P1   120.0000
 P2   150.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              4290.000
Objective bound:                              4290.000
Infeasibilities:                              0.000000

OPTIMAL PRODUCTION PROGRAM:
P1:   12un  x  Unit Profit: $120.00  =  Total: $1440.00
P2:   19un  x  Unit Profit: $150.00  =  Total: $2850.00
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

Two factories (A, B) produce 3 different types of paper. The company that controls both has a contract to produce:

Products / Resource			A	B	Available
Paper	Thin	ton	8	2	16
	Medium	ton	1	1	6
	Thick	ton	2	7	28
Unit Cost		$	1000.00	2000.00	-

How many days will each plant have to operate to supply the orders more economically?

```
MODEL:
SETS:
 PRODUCT: DEMAND;
 RESOURCE: COST;
 RXP( PRODUCT, RESOURCE) : CAPACITY, VOLUME;
ENDSETS
DATA:
! Products attributes;
 PRODUCT,          DEMAND       =
 THIN              16
 MEDIUM            6
 THICK             28;
! Resources attributes;
 RESOURCE,         COST         =
 PLANT_A           1800
 PLANT_B           2000;
! Daily production  Plant_A      Plant_B;
 CAPACITY    =     8            2              !Thin;
                   1            1              !Medium;
                   2            7;             !Thick;
ENDDATA
SUBMODEL MIN5:
[OBJ] MIN = @SUM( RXP(r, p): COST( p) * VOLUME( r, p));
! The Demand constraints;
@FOR( PRODUCT( L):
        [DEM] @SUM (RESOURCE(C): VOLUME( L, C)) = DEMAND ( L););
! The Capacity constraints;
@FOR( RXP( r, p):
        [CAP] @SUM( RXP( r,p ): VOLUME( r,p )) >= CAPACITY( r, p););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1) ;
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  DAILY PRODUCTION (TON)  :", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND:", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) : '  Product:',
        @FORMAT(PRODUCT(I),'-7s'), ' Produce: ',
        @FORMAT(VOLUME(I,J),'%5.1f'),"ton ", '(in ' ,
        @FORMAT(VOLUME(I,J)/CAPACITY(I,J),'%4.1f'), ' Days) ',
        @FORMAT(RESOURCE(J),"-6s"), ' x ', 'Unit cost: $',
        @FORMAT(COST( J),'%6.2f'), ' = ', 'Total: $',
        @FORMAT(VOLUME(I, J) * COST( J),'%8.2f'),
@NEWLINE( 1));
! Demand reached;
@WRITE('  Demand reached in:',20*' ',
@FORMAT(@MAX( RXP(I, J):VOLUME/CAPACITY),'%4.1f'),' Days' , @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
DAILY PRODUCTION (TON)  :
          PLANT_A   PLANT_B
 THIN    8.000000  2.000000
 MEDIUM  1.000000  1.000000
 THICK   2.000000  7.000000

DEMAND:
 THIN    16.00000
 MEDIUM  6.000000
 THICK   28.00000

COST:
 PLANT_A  1000.000
 PLANT_B  2000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                                                   92000.00
Infeasibilities:                                                                                   0.000000

OPTIMAL PRODUCTION PROGRAM:
Product:THIN    Produce:  14.0ton  (in  1.8 Days) PLANT_A x Unit cost: $1800.00 = Total: $25200.00
Product:THIN    Produce:   2.0ton  (in  1.0 Days) PLANT_B x Unit cost: $2000.00 = Total: $ 4000.00
Product:MEDIUM  Produce:   5.0ton  (in  5.0 Days) PLANT_A x Unit cost: $1800.00 = Total: $ 9000.00
Product:MEDIUM  Produce:   1.0ton  (in  1.0 Days) PLANT_B x Unit cost: $2000.00 = Total: $ 2000.00
Product:THICK   Produce:  21.0ton  (in 10.5 Days) PLANT_A x Unit cost: $1800.00 = Total: $37800.00
Product:THICK   Produce:   7.0ton  (in  1.0 Days) PLANT_B x Unit cost: $2000.00 = Total: $14000.00
Demand reached in:                    10.5 Days
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

GOAL

One company produces parachutes and hang gliders on two assembly lines.

Resources / Products			Parachutes	Hang Gliders	Available
Assembler	Line A	hr	10	10	100
	Line B	hr	3	7	42
Profit		$	60.00	40.00	-

Formulate the production schedule that maximize the profit of the Company?

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE ,              AVAILABLE   =
 LINE_A                  100
 LINE_B                  42;

! Products attributes;
 PRODUCT,                PROFIT      =
 PARACHUTES              60
 HANG_GLIDERS            40;

!  Required ( hr)        Parachutes  HangGliders;
 USAGE     =             10          10          ! LINE_A;
                         3           7;          ! LINE_B;
ENDDATA
SUBMODEL MAX6:
MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   REQUIRED:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX6);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) | USAGE( I,J) #GT# 0 #AND# USAGE(I, J) #LT# 4: '  ',
      @FORMAT(RESOURCE(I),'-6s'), ', Produce:',
      @FORMAT(PRODUCE( J),'%2.0f'),' ',
      @FORMAT(PRODUCT(J),'-12s'), 'x Unit profit: $',
      @FORMAT(PROFIT( J),'%4.2f'), ' = ', 'Total: $',
      @FORMAT(PRODUCE( J) * PROFIT( J),'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (hr):
             PARACHUTES  HANG_GLIDERS
 LINE_A        10.00000      10.00000
 LINE_B        3.000000      7.000000

AVAILABLE (hr/week):
 LINE_A        100.0000
 LINE_B        42.00000

PROFIT (p/un):
 PARACHUTES    60.00000
 HANG_GLIDERS  40.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                       600.0000
Infeasibilities:                                       0.000000

OPTIMAL PRODUCTION PROGRAM:
LINE_B, Produce:10 PARACHUTES  x Unit profit: $60.00 = Total: $600.00
```

7

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

One factory intends to manufacture two models of suits and has the following stock:

Resources / Products			Mod1	Mod2	Available
Raw Material	Jeans	m	4	2	32
	Silk	m	2	4	22
	Satin	m	2	6	30
Price		$	6.00	10.00	-

How many pieces of each type the manufacturer must make to get the maximum revenue.

```
MODEL:
SETS:
 PRODUCT : PRICE, PRODUCE, IND;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE ,        AVAILABLE    =
 JEANS             32
 SILK              22
 SATIN             30;

! Product attributes;
 PRODUCT,          PRICE,       IND     =
 MOD1              6            0
 MOD2              10           1;

 ! Required (m)    MOD1         MOD2;
 USAGE =           4            2            ! JEANS;
                   2            4            ! SILK;
                   2            6;           ! SATIN;
ENDDATA
SUBMODEL MAX7:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (m):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (m):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX7);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) : '  ',
        @FORMAT(PRODUCT(J), '-5s'),'Produce:',
        @FORMAT(PRODUCE( J),'%2g'),'un x ', 'Unit price:$',
        @FORMAT(PRICE( J),'%5.2f'), ' = ', 'Revenue:$',
        @FORMAT(PRODUCE( J) * PRICE( J),'%5.2f'),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (m):
           MOD1      MOD2
 JEANS  4.000000  2.000000
 SILK   2.000000  4.000000
 SATIN  2.000000  6.000000

AVAILABLE (m):
 JEANS  32.00000
 SILK   22.00000
 SATIN  30.00000

PRICE:
 MOD1  6.000000
 MOD2  10.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
SOLUTION
Global optimal solution found.
Objective value:                              62.00000
Infeasibilities:                              0.000000

OPTIMAL PRODUCTION PROGRAM:
MOD1 Produce: 7un x Unit price:$ 6.00 = Revenue:$42.00
MOD2 Produce: 2un x Unit price:$10.00 = Revenue:$20.00
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

A leather goods company, manufactures two types of products: bags and backpacks.

Resources / Products			Bags	Backpack	Available
Process	Cutting	hr	2	0	300
	Seam	hr	2	2	440
	Packing	hr	1.2	1.5	300
	Dyeing	hr	0	3	540
Production Day		un	120	30	150
Profit		$	50.00	40.00	-

Determines how many of each product the company must manufacture to maximize its profit.

```
MODEL:
SETS:
 PRODUCT : PROFIT, DEMAND, IND,  PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
RESOURCE,   AVAILABLE =
CUTTING     300
SEAM        440
PACKING     300
DEYING      540;
! Products attributes;
PRODUCT,    DEMAND,     PROFIT,     IND =
BAGS        120         50          0
BACKPACkS   30          40          1;
! Required  BAGS        BACKPACkS;
USAGE =     2           0           ! CUTTING;
            2           2           ! SEAM;
            1.2         1.5         ! PACKING;
            0           3;          ! DEYING;
ENDDATA
SUBMODEL MAX8:
MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Demand constraints;
@FOR( PRODUCT( J): [DEM] @SUM( PRODUCT( J): PRODUCE( J)) = DEMAND( J));
! The Available constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (hr/un):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (hr):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/un):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX8);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) : '  ',
        @FORMAT(PRODUCT(J),'-10s'),' Produce:',
        @FORMAT(PRODUCE( J),'%3.0f'),'un x ', 'Profit:$',
        @FORMAT(PROFIT( J),'%5.2f'), ' = ', 'Total:$',
        @FORMAT(PRODUCE( J) * PROFIT( J),'%6.2f'),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX8);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (hr/un):
               BAGS   BACKPACKS
 CUTTING   2.000000   0.000000
 SEAM      2.000000   2.000000
 PACKING   1.200000   1.500000
 DEYING    0.000000   3.000000

AVAILABLE (hr):
 CUTTING  300.0000
 SEAM   440.0000
 PACKING  300.0000
 DEYING   540.0000

DEMAND (un):
 BAGS       120.0000
 BACKPACKS  30.00000

PROFIT (p/un):
 BAGS       50.00000
 BACKPACKS  40.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                    7200.000
Infeasibilities:                                    0.000000

OPTIMAL PRODUCTION PROGRAM:
BAGS       Produce:120un x Profit:$50.00 = Total:$6000.00
BACKPACKS  Produce: 30un x Profit:$40.00 = Total:$1200.00
```

9

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

A manufacturing company produces metal structures for two categories of bicycles. All the necessary information for the development of the model follows below:

Resources / Products			Mountain	Racing	Available
Process	Frame	min	25	10	19,000
	Paint	min	10	15	18,000
	Quality	min	10	25	15,000
Fixed Cost		$	162,559.00	92,441.00	-
Price		$	800.00	1000.00	-
Minimum Production		un	250	1	-

Due to the responsibilities of the contractors there is a goal of producing over 250 Mountain Bike per month. What is the Best Monthly Production Plan?

```
MODEL:
SETS:
  PRODUCT : PRICE, COST, FCOST, MIN_PROD, IND, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE,              AVAILABLE  =
  FRAME                  19000
  PAINT                  18000
  QUALITY                15000;
! Products attributes;
  PRODUCT,               PRICE,     COST,      MIN_PROD,  FCOST      IND  =
  MOUNTAIN               800        300        250        162559     0
  RACING                 1000       400        1          92441      1;
! Required               MOUNTAIN   RACING;
  USAGE      =           10         25         ! Frame;
                         18         15         ! Paint;
                         10         25;        ! Quality;
ENDDATA
SUBMODEL MAX9:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p) - COST( P) * PRODUCE(P) - FCOST( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( P): usage( R,P) * PRODUCE( P )) <= AVAILABLE( R); @GIN( PRODUCE));
! Minimum production constraints;
@FOR( PRODUCT( P): PRODUCE(P) >= MIN_PROD(P));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;

@SET('TERSEO',1);

! Post status windows, 1 Yes, 0 No;

@SET('STAWIN',0);

! Data block;

@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (Minutes):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Minutes):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (unity):", @NEWLINE( 1));
@TABLE(MIN_PROD);
@WRITE(" ", @NEWLINE( 1), "  FIXED COST:", @NEWLINE( 1));
@TABLE(FCOST);
@WRITE(" ", @NEWLINE( 1), "  UNIT PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX9);
! Solution report;
@WRITE("  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J):'  ',
      @FORMAT(PRODUCT(J),'-8s'),' Produce:',
      @FORMAT(PRODUCE( J),'%3.0f'), ' x  cost: $',
      @FORMAT(COST( J),'%7.2f'), ' =  total:$',
      @FORMAT(COST( J) * PRODUCE( J), '%8.2f'), ' - ',@NEWLINE(1),'  ',37*' ', 'Fixed cost:$',
      @FORMAT(FCOST(J),'%9.2f'), ' - ', @NEWLINE(1), '  ',
      @FORMAT(PRODUCT(J),'-8s'),' Produce:',
      @FORMAT(PRODUCE( J),'%3.0f'), ' x  price:$',
      @FORMAT(PRICE( J),'%7.2f'),' = ', ' Total:$',
      @FORMAT(PRODUCE( J) * Price( J),'%7.2f' ),' + ',  @NEWLINE(1),'  ', 'PROFIT:', 41*' ','$',
      @FORMAT(PRODUCE( J) * PRICE( J) - COST(J)  * PRODUCE( J) - FCOST(J),'%9.2f'),' = ',
@NEWLINE( 2));
@WRITE('  TOTAL PROFIT:', 35*' ','$', @FORMAT(OBJ,'%9.2f'), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX9);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (Minutes):
          MOUNTAIN    RACING
 FRAME    10.00000  25.00000
 PAINT    18.00000  15.00000
 QUALITY  10.00000  25.00000

AVAILABLE (Minutes):
 FRAME    19000.00
 PAINT    18000.00
 QUALITY  15000.00
```

```
MINIMUM PRODUCTION (unity):
 MOUNTAIN  250.0000
 RACING    1.000000

FIXED COST:
 MOUNTAIN  162559.0
 RACING    92441.00

UNIT PRICE:
 MOUNTAIN  800.0000
 RACING    1000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                        300000.0
Objective bound:                                        300000.0
Infeasibilities:                                        0.000000

OPTIMAL PRODUCTION PROGRAM:
MOUNTAIN Produce:750 x  cost: $ 300.00 =  total:$225000.00 -
                                    Fixed cost:$162559.00 -
MOUNTAIN Produce:750 x  price:$ 800.00 =  Total:$600000.00 +
PROFIT:                                        $212441.00 =

RACING   Produce:300 x  cost: $ 400.00 =  total:$120000.00 -
                                    Fixed cost:$ 92441.00 -
RACING   Produce:300 x  price:$1000.00 =  Total:$300000.00 +
PROFIT:                                        $ 87559.00 =

TOTAL PROFIT:                                  $300000.00
```

10

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

GOAL

One manufacturer is starting the last week of production of four different models of wooden consoles for television sets designated M1, M2, M3 and M4. Each of them should be assembled and then decorated.

The manufacturer has:

- 30,000 hours to assemble these products (750 assemblers working 40 hours per week)
- 20,000 hours to decorate these products (500 decorators working 40 hours per week).

All the necessary information for the development of the model follows below:

Resources / Products			M1	M2	M3	M4	Available	Workers	HR/W
Process	Assembler	hr	4	5	3	5	30,000	750	40
	Decor	hr	2	1	5	3	20,000	500	40
Profit		$	7.00	7.00	6.00	9.00	-	-	-

How much of each model should be produced during this last week to maximize profit?

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,                     AVAILABLE   =
 ASSEMBLER                     30000
 DECOR                         20000;
! Product attributes;
 PRODUCT ,                     PROFIT      =
 M1                            7
 M2                            7
 M3                            6
 M4                            9;

! Hours required per unit      M1    M2    M3    M4;
 USAGE       =                 4     5     3     5     ! Assembler;
                               2     1     5     3;    ! Decor;
ENDDATA
SUBMODEL MAX10:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r);
        @GIN( PRODUCE );); 
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (hour):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (hour):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/un):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX10);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE( J) #GT# 0: '  ', PRODUCT(J),', Produce:',
        @FORMAT(PRODUCE( J),'%5.0f'),' un x ', 'Unit price: $',
        @FORMAT(PROFIT( J),'%4.2f'),' = ', 'Total: $',
        @FORMAT(PRODUCE( J) * PROFIT( J),'%8.2f' ),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
! @GEN(MAX10);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (hour):
                  M1        M2        M3        M4
 ASSEMBLER  4.000000  5.000000  3.000000  5.000000
 DECOR      2.000000  1.000000  5.000000  3.000000

AVAILABLE (hour):
 ASSEMBLER  30000.00
 DECOR      20000.00

PROFIT (p/un):
 M1         7.000000
 M2         7.000000
 M3         6.000000
 M4         9.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                   54375.00
Objective bound:                                   54375.00
Infeasibilities:                                   0.000000

OPTIMAL PRODUCTION PROGRAM:
M3, Produce:  625 un x Unit price: $6.00 = Total: $ 3750.00
M4, Produce: 5625 un x Unit price: $9.00 = Total: $50625.00
```

11

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Purchase

Source:

Book 8

Chapter 1.2.5

GOAL

One company manufactures motors for toys and small appliances. The Marketing department is forecasting sales of 6100 RONCAM engine units in the next six months.

This is a good demand and the company will have to test its productive capacity. All the necessary information for the development of the model follows below:

Resources / Components			Made			Purchase			Available
			Body	Base	Blind	Body	Base	Blind	
Process	Preparation	UT	2	5	4				49,200
	Mold	UT	4	2	5				49,200
	Assembler	UT	2	4	5				49,200
Unit Cost		$	8.00	20.00	10.00	10.00	20.00	16.00	-

Some of these components may be purchased from other suppliers if there are company limitations. Based on this information minimize the cost of the company.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources  attributes UT (Unit of time);
 RESOURCE ,             AVAILABLE   =
 PREPARATION            49200
 MOLD                   49200
 ASSEMBLER              49200;

! Products attributes;
 PRODUCT ,              COST        =
 BODY                   8
 BASE                   20
 BLIND                  10
 BODY_P                 10
 BASE_P                 20
 BLIND_P                16;

! Hours required p/unit Body  Base  Blind  Body_P  Base_P  Blind_P;
 USAGE      =           2     5     4      0       0       0         ! Preparation (UT (Unit of time);
                        4     2     5      0       0       0         ! Mold (UT (Unit of time);
                        2     4     5      0       0       0;        ! Assembler (UT (Unit of time);
ENDDATA
SUBMODEL MIN11:
MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r); );
! forecasting sales;
PRODUCE(1) + PRODUCE(4) >= 6100;
PRODUCE(2) + PRODUCE(5) >= 6100;
PRODUCE(3) + PRODUCE(6) >= 6100;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (UT):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (UT):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN11);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION/PURCHASE PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '   ',
        @FORMAT(PRODUCT(J),'-7s'), @IF(@STRLEN(PRODUCT(J)) #GT# 5, '  Purchase:', '  Produce :'),
        @FORMAT(PRODUCE( J),'%5.0f'),' un x ', 'Unit cost: $',
        @FORMAT(COST( J),'%5.2f'),' = ', 'Total: $',
        @FORMAT(produce( J) * COST( J),'%9.2f' ),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN11);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (UT – Unit of Time):
                   BODY      BASE      BLIND     BODY_P    BASE_P    BLIND_P
 PREPARATION  2.000000  5.000000  4.000000  0.000000  0.000000  0.000000
 MOLD         4.000000  2.000000  5.000000  0.000000  0.000000  0.000000
 ASSEMBLER    2.000000  4.000000  5.000000  0.000000  0.000000  0.000000

AVAILABLE (UT):
 PREPARATION  49200.00
 MOLD         49200.00
 ASSEMBLER    49200.00

COST:
 BODY         8.000000
 BASE         20.00000
 BLIND        10.00000
 BODY_P       10.00000
 BASE_P       20.00000
 BLIND_P      16.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                         234650.0
Infeasibilities:                                         0.000000

OPTIMAL PRODUCTION/PURCHASE PROGRAM:
BODY   Produce : 4675 un x Unit cost: $ 8.00 = Total: $ 37400.00
BLIND  Produce : 6100 un x Unit cost: $10.00 = Total: $ 61000.00
BODY_P Purchase: 1425 un x Unit cost: $10.00 = Total: $ 14250.00
BASE_P Purchase: 6100 un x Unit cost: $20.00 = Total: $122000.00
```

12

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 227

GOAL

An electronics company manufactures five different models of communication cards for microcomputers.

Here are the materials and component quantities that make up each card model:

Resources / Products			H1	F1	V1	M1	M2	Available
Raw Material	Printed Circuit	m2	20	15	10	8	5	80,000
	Resistors	un	28	24	18	12	16	100,000
	Memory	un	8	8	4	4	6	30,000
	Assembler	hr	0.75	0.6	0.5	0.65	1	5,000
Minimum Production		un	500	1,000	500	500	500	-
Cost		$	136.00	101.00	96.00	137.00	101.00	-
Price		$	189.00	149.00	129.00	169.00	139.00	-

The company can sell every quantity produced, however, the marketing department wants to produce at least 500 units of each card model.

This department also determines that the quantity produced from the F1 card is at least twice the quantity sold from the H1 carton.

Based on this information maximize the profit of the company.

```
MODEL:
SETS:
  PRODUCT :PRICE,  COST, PRODUCE, MIN_PROD;
  HEADER / H1, F1, V1, M1, M2, AVAIL, VALUE /:;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
  PXR( RESOURCE,HEADER): SLASUR;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE,           AVAILABLE   =
  PRT_CIRCUIT_M2      80000
  RESISTORS_UN        100000
  MEMORY_UN           30000
  ASSEMBLER_HR        5000;
! Products attributes;
  PRODUCT,            PRICE,      COST,       MIN_PROD  =
  H1                  189         136         500
  F1                  149         101         1000
  V1                  129         96          500
  M1                  169         137         500
  M2                  139         101         500;

! Required per unit   H1          F1          V1          M1          M2;
  USAGE       =       20          15          10          8           5      ! PRT_CIRCUIT_M2;
                      28          24          18          12          16     ! RESISTORS_UN;
                      8           8           4           4           6      ! MEMORY_UN;
                      0.75        0.6         0.5         0.65        1;     ! ASSEMBLER_HR;
ENDDATA
SUBMODEL MAX12:
MAX = @SUM( PRODUCT( p): PRICE(p) * produce(p) - COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
! Minimum production required;
@FOR( PRODUCT( p): PRODUCE(p) >= MIN_PROD(p));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Sets the length of the line;
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (m2, un, un, hr):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (m2, un, un, hr)::", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (un):", @NEWLINE( 1));
@TABLE(MIN_PROD);
@WRITE(" ", @NEWLINE( 1), "  COST p/un:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  PRICE p/un:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX12);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE( J) #GT# 0: '   ',
      @FORMAT(PRODUCT(J),'-3s'),'Produce:',
      @FORMAT(PRODUCE( J),'%4.0f'),'un x ', 'Price:$',
      @FORMAT(PRICE( J),'%4.2f'),' = ', 'Revenue:$',
      @FORMAT(PRODUCE( J) * PRICE( J),'%9.2f' ),' - Cost:$',
      @FORMAT(PRODUCE( J) * COST( J),'%9.2f' ),' = Profit:$',
      @FORMAT(PRODUCE( J) * PRICE( J) - PRODUCE( J) * COST( J),'%7.2f' ),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
! Slack / Surplus report;
@WRITE("  SLACK/SURPLUS LIMIT = AVAILABLE: " , @NEWLINE( 1));
@FOR(PXR(I,J): SLASUR(I,6) = AVAILABLE(I));
@FOR(RXP(I,J)| J #LT# 6: SLASUR(I,J) = USAGE(I,J) * PRODUCE(J));
@FOR(PXR(I,J): SLASUR(I,7) = SLASUR(I,1) + SLASUR(I,2) + SLASUR(I,3) + SLASUR(I,4) + SLASUR(I,5)- SLASUR(I,6));
@TABLE(SLASUR);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX12);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (m2, un, un, hr):
                       H1         F1         V1         M1         M2
 PRT_CIRCUIT_M2   20.00000   15.00000   10.00000   8.000000   5.000000
 RESISTORS_UN     28.00000   24.00000   18.00000   12.00000   16.00000
 MEMORY_UN        8.000000   8.000000   4.000000   4.000000   6.000000
 ASSEMBLER_HR     0.750000   0.600000   0.500000   0.650000   1.000000

AVAILABLE (m2, un, un, hr):
 PRT_CIRCUIT_M2   80000.00
 RESISTORS_UN     100000.0
 MEMORY_UN        30000.00
 ASSEMBLER_HR     5000.000

MINIMUM PRODUCTION (un):     COST p/un:              PRICE p/un:
 H1   500.0000                H1   136.0000           H1   189.0000
 F1   1000.000                F1   101.0000           F1   149.0000
 V1   500.0000                V1   96.00000           V1   129.0000
 M1   500.0000                M1   137.0000           M1   169.0000
 M2   500.0000                M2   101.0000           M2   139.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                    215000.0
Objective bound:                                                    215000.0
Infeasibilities:                                                    0.000000

Model Class: LP

OPTIMAL PRODUCTION PROGRAM:
 H1 Produce: 500un x Price:$189.00 = Revenue:$ 94500.00 - Cost:$ 68000.00 = Profit:$26500.00
 F1 Produce:1000un x Price:$149.00 = Revenue:$149000.00 - Cost:$101000.00 = Profit:$48000.00
 V1 Produce:1500un x Price:$129.00 = Revenue:$193500.00 - Cost:$144000.00 = Profit:$49500.00
 M1 Produce:2250un x Price:$169.00 = Revenue:$380250.00 - Cost:$308250.00 = Profit:$72000.00
 M2 Produce: 500un x Price:$139.00 = Revenue:$ 69500.00 - Cost:$ 50500.00 = Profit:$19000.00

SLACK/SURPLUS LIMIT = AVAILABLE:
                      H1         F1         V1         M1         M2      LIMIT      VALUE
 PRT_CIRCUIT_M2   10000.00   15000.00   15000.00   18000.00   2500.000   80000.00  -19500.00
 RESISTORS_UN     14000.00   24000.00   27000.00   27000.00   8000.000   100000.0   0.000000
 MEMORY_UN        4000.000   8000.000   6000.000   9000.000   3000.000   30000.00   0.000000
 ASSEMBLER_HR     375.0000   600.0000   750.0000   1462.500   500.0000   5000.000  -1312.500
```

13

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

In a product mix model, the decision is how many different products should be produced to maximize total profit. All the information needed to develop the model follows:

Resources / Products			Rocket	Meteor	Streak	Comet	Jet	Biplane	Available
Raw Material	Steel	un	1	4	0	4	2	0	800
	Cooper	un	4	5	3	0	1	0	1,160
	Plastic	un	0	3	8	0	1	0	1,780
	Rubber	un	2	0	1	2	1	5	1,050
	Glass	un	2	4	2	2	2	4	1,360
	Paint	un	1	4	1	4	3	4	1,240
Minimum Production		un	400	400	400	400	400	400	-
Setup		$	35.00	20.00	60.00	70.00	75.00	30.00	-
Profit		$	30.00	45.00	24.00	26.00	24.00	30.00	-

Each of the products compete with several scarce resources. In this example, we produce six different flying machines from six different raw materials.

This model also has the characteristic that, if we want to produce a specific product, we assume a fixed installation cost.

```
MODEL:
SETS:
 PRODUCT: PROFIT, SETUP, PRODUCE, MIN_PROD;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Product Attributes;
PRODUCT,        PROFIT,     SETUP      MIN_PROD          =
ROCKET          30          35         400
METEOR          45          20         400
STREAK          24          60         400
COMET           26          70         400
JET             24          75         400
BIPLANE         30          30         400;

! Resources attributes;
RESOURCE,       AVAILABLE    =
STEEL           800
COPPER          1160
PLASTIC         1780
RUBBER          1050
GLASS           1360
PAINT           1240;

!  Required p/unit (hr)  Rocket Meteor Streak  Comet Jet    Biplane;
USAGE        =          1      4      0       4     2      0          ! STEEL;
                        4      5      3       0     1      0          ! COPPER;
                        0      3      8       0     1      0          ! PLASTIC;
                        2      0      1       2     1      5          ! RUBBER;
                        2      4      2       2     2      4          ! GLASS;
                        1      4      1       4     3      4;         ! PAINT;
ENDDATA
SUBMODEL MAX13:
[OBJ] MAX = @SUM( PRODUCT: PROFIT * PRODUCE - SETUP );
! The Available constraints;
@FOR( RESOURCE( I):
          [AVA]  @SUM( PRODUCT( J):  USAGE( I, J) * PRODUCE( J)) <= AVAILABLE( I) );
@FOR( PRODUCT: @GIN( PRODUCE));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Sets the length of the line;
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (un):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (un):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (un):", @NEWLINE( 1));
@TABLE(MIN_PROD);
@WRITE(" ", @NEWLINE( 1), "  SETUP:", @NEWLINE( 1));
@TABLE(SETUP);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX13);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
          @FORMAT(PRODUCT(J),'-8s'),' Produce:',
          @FORMAT(PRODUCE( J),'%3.0f'),' un x ', 'Price: $',
          @FORMAT(PROFIT( J),'%4.2f'),' = ', 'Revenue: $',
          @FORMAT(PRODUCE( J) * PROFIT( J),'%8.2f' ),' - Setup: $',
          @FORMAT(SETUP( J),'%5.2f' ),' = Profit: $',
          @FORMAT(PRODUCE( J) * PROFIT( J) - SETUP( J) ,'%8.2f' ),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX13);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (un):
            ROCKET    METEOR    STREAK     COMET       JET   BIPLANE
 STEEL    1.000000  4.000000  0.000000  4.000000  2.000000  0.000000
 COPPER   4.000000  5.000000  3.000000  0.000000  1.000000  0.000000
 PLASTIC  0.000000  3.000000  8.000000  0.000000  1.000000  0.000000
 RUBBER   2.000000  0.000000  1.000000  2.000000  1.000000  5.000000
 GLASS    2.000000  4.000000  2.000000  2.000000  2.000000  4.000000
 PAINT    1.000000  4.000000  1.000000  4.000000  3.000000  4.000000

AVAILABLE (un):                 SETUP (p/product):
 STEEL    800.0000               ROCKET   35.00000
 COPPER   1160.000               METEOR   20.00000
 PLASTIC  1780.000               STREAK   60.00000
 RUBBER   1050.000               COMET    70.00000
 GLASS    1360.000               JET      75.00000
 PAINT    1240.000               BIPLANE  30.00000

MINIMUM PRODUCTION (un) :       PROFIT p/un:
 ROCKET   400.0000               ROCKET   30.00000
 METEOR   400.0000               METEOR   45.00000
 STREAK   400.0000               STREAK   24.00000
 COMET    400.0000               COMET    26.00000
 JET      400.0000               JET      24.00000
 BIPLANE  400.0000               BIPLANE  30.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                      14730.00
Objective bound:                                                      14730.00
Infeasibilities:                                                      0.000000

OPTIMAL PRODUCTION PROGRAM:
ROCKET   Produce:120 un x Price: $30.00 = Revenue: $ 3600.00 - Setup: $35.00 = Profit: $ 3565.00
STREAK   Produce:220 un x Price: $24.00 = Revenue: $ 5280.00 - Setup: $60.00 = Profit: $ 5220.00
COMET    Produce:160 un x Price: $26.00 = Revenue: $ 4160.00 - Setup: $70.00 = Profit: $ 4090.00
JET      Produce: 20 un x Price: $24.00 = Revenue: $  480.00 - Setup: $75.00 = Profit: $  405.00
BIPLANE  Produce: 50 un x Price: $30.00 = Revenue: $ 1500.00 - Setup: $30.00 = Profit: $ 1470.00
```

14

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

Source:

- Book 1
- Page 188

GOAL

In the manufacture of certain products, the following raw materials are used in the quantities indicated below:

Resources / Products			PA	PB	PC	PD	Available
Raw Material	Wood	Ton	2	1	3	0	1,000
	Plastic	Ton	0	4	1	0	2,000
	Steel	Ton	1	0	3	2	800
	Glass	Ton	0	0	2	2	1,000
	Ink	Ton	0	1	2	1	1,500
Profit		$	20.00	30.00	25.00	15.00	-

Raw material stocks in tons are in the table above as availability and profits in the last line. We ask for the maximum profit scheme.

```
MODEL:
SETS:
  PRODUCT : PROFIT, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE ,        AVAILABLE    =
  WOOD              1000
  PLASTIC           2000
  STEEL             800
  GLASS             1000
  INK               1500;
! Products attributes;
  PRODUCT ,         PROFIT        =
  PA                20
  PB                30
  PC                25
  PD                15;
! Required          PA    PB    PC    PD;
  USAGE     =       2     1     3     0     ! WOOD;
                    0     4     1     0     ! PLASTIC;
                    1     0     3     2     ! STEEL;
                    0     0     2     2     ! GLASS;
                    0     1     2     1;    ! INK;
ENDDATA
SUBMODEL MIN14:
MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (Ton):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Ton):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/Ton):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN14);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION/PURCHASE PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE(J) #GT# 0: '  ',
        @FORMAT(PRODUCT(J),'-2s'),' Produced:',
        @FORMAT(PRODUCE( J),'%4.0f'),'ton x ', 'Profit: $',
        @FORMAT(PROFIT( J),'%4.2f'),' = ', 'Total: $',
        @FORMAT(PRODUCE( J) * PROFIT( J),'%8.2f' ),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN14);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (Ton):
                 PA         PB         PC         PD
 WOOD      2.000000  1.000000  3.000000  0.000000
 PLASTIC   0.000000  4.000000  1.000000  0.000000
 STEEL     1.000000  0.000000  3.000000  2.000000
 GLASS     0.000000  0.000000  2.000000  2.000000
 INK       0.000000  1.000000  2.000000  1.000000

AVAILABLE (Ton):
 WOOD      1000.000
 PLASTIC   2000.000
 STEEL     800.0000
 GLASS     1000.000
 INK       1500.000

PROFIT (p/Ton):
 PA  20.00000
 PB  30.00000
 PC  25.00000
 PD  15.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                             24125.00
Infeasibilities:                             0.000000

OPTIMAL PRODUCTION/PURCHASE PROGRAM:
PA Produced: 250ton x Profit: $20.00 = Total: $ 5000.00
PB Produced: 500ton x Profit: $30.00 = Total: $15000.00
PD Produced: 275ton x Profit: $15.00 = Total: $ 4125.00
```

15

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

GOAL

One company is planning its next production cycle, to produce three products. For this they must go through three productive processes and consume different amount of hours as follows:

Resources / Products			P1	P2	P3	Available
Process	Machining	hr	2	3	6	600
	Polymer	hr	6	3	4	300
	Assembler	hr	5	6	2	400
Minimum Production		un	50	67	75	-
Setup		$	1000	800	900	-
Profit		$	48.00	55.00	50.00	-

The production manager wants to determine the most profitable production mix.

```
MODEL:
SETS:
 PRODUCT: PROFIT, SETUP, PRODUCE, MIN_PROD;
 RESOURCE : AVAILABLE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Product Attributes;
PRODUCT,          PROFIT,     SETUP,      MIN_PROD =
PROD1             48          1000        50
PROD2             55          800         67
PROD3             50          900         75;
! Available resources;
RESOURCE,         AVAILABLE =
MACHINING         600
POLYMER           300
ASSEMBLER         400;

!  Required       Machining   Polymer     Assembler;
USAGE   =         2           3           6           ! PROD1;
                  6           3           4           ! PROD2;
                  5           6           2;          ! PROD3;
ENDDATA
SUBMODEL MAX15:
[OBJ] MAX = @SUM( PRODUCT(I): PROFIT(I) * PRODUCE(I) - @IF(PRODUCE #GT# 0, SETUP(I),0));
! The Available constraints;
@FOR( RESOURCE( I):
        [AVA] @SUM( PRODUCT( J):  USAGE( I, J) * PRODUCE( J)) <= AVAILABLE( I) );
! Minimum production required;
@FOR( PRODUCT:[PROD] PRODUCE <= MIN_PROD );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Sets the length of the line;
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE (hr):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (hr):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (un):", @NEWLINE( 1));
@TABLE(MIN_PROD);
@WRITE(" ", @NEWLINE( 1), "  SETUP (p/product):", @NEWLINE( 1));
@TABLE(SETUP);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/un):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX15);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ', PRODUCT(J),' Produce:',
        @FORMAT(PRODUCE( J),'%3.0f'),' un x ', 'Unity profit: $',
        @FORMAT(PROFIT( J),'%4.2f'),' = ', 'Total: $',
        @FORMAT(PRODUCE( J) * PROFIT( J),'%6.2f' ),' - Setup: $',
        @FORMAT(SETUP( J),'%4.2f' ),' = Profit: $',
        @FORMAT(PRODUCE( J) * PROFIT( J) - SETUP( J) ,'%7.2f' ),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX15);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (hour):
              PROD1     PROD2     PROD3
 MACHINING  2.000000  3.000000  6.000000
 POLYMER    6.000000  3.000000  4.000000
 ASSEMBLER  5.000000  6.000000  2.000000

AVAILABLE:
 MACHINING  600.0000
 POLYMER    300.0000
 ASSEMBLER  400.0000

MINIMUM PRODUCTION:
 PROD1      50.00000
 PROD2      67.00000
 PROD3      75.00000

SETUP:
 PROD1      1000.000
 PROD2      800.0000
 PROD3      900.0000

PROFIT:
 PROD1      48.00000
 PROD2      55.00000
 PROD3      50.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                                            3022.22
Objective bound:                                                                            3022.22
Infeasibilities:                                                                            0.00000

OPTIMAL PRODUCTION PROGRAM:
PROD2 Produce: 56 un x Unity profit: $55.00 = Total: $3055.56 - Setup: $800.00 = Profit: $2255.56
PROD3 Produce: 33 un x Unity profit: $50.00 = Total: $1666.67 - Setup: $900.00 = Profit: $ 766.67
```

16

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities

Source:

- Book 2
- Page 228

GOAL

A company has three manufacturing facilities that can each produce three different products. All the necessary information for the development of the model follows below:

Resources / Products			Prod1	Prod2	Prod3	Available
Facilities	INST_1	un	8	4	10	800
	INST_2	un	9	6	10	1,000
	INST_3	un	12	10	11	1,200
Demand		un	1,000	900	800	-
Price		$	20.00	25.00	30.00	-

What is the best production plan to maximize profit?

```
MODEL:
SETS:
 PRODUCT : PRICE, DEMAND, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) :  USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        AVAILABLE =
 INST1            800
 INST2            1000
 INST3            1200;

! Product attributes;
 PRODUCT,         PRICE,      DEMAND =
 PROD1            20          1000
 PROD2            25          900
 PROD3            30          800;

 !Required        PROD1            PROD2            PROD3;
 USAGE =          8                4                10            ! INST1;
                  9                6                10            ! INST2;
                  12               10               11 ;          ! INST3;
ENDDATA
SUBMODEL MAX16:
[OBJ] MAX = XPRICE - XCOST ;
      XPRICE = @SUM( RXP (r, p): PRODUCE( p) * price( p));
      XCOST  = @SUM( RXP( r, p): PRODUCE( p) * usage( r, p));
! The Available constraints;
@FOR( PRODUCT( J):
      [AVA] @SUM( PRODUCT(c): PRODUCE( C)) <= AVAILABLE( j));
! The Demand constraints;
@FOR( RESOURCE( k):
 [DEM] @SUM( RESOURCE( p): PRODUCE( p) ) <= DEMAND( k); );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/PRODUCT (un):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (un):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  PRICE (p/un):", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1));
@WRITE("  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX16);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) | USAGE( I,J) * PRODUCE(J) #GT# 0: '  ',
      @FORMAT(RESOURCE(I),'-6s'), ' ', PRODUCT(J),' Produce: ',
      @FORMAT(PRODUCE( J),'%3.0f'),' un x ', 'Price: $',
      @FORMAT(PRICE( J),'%5.2f'), ' = ', 'Total: $',
      @FORMAT(PRODUCE(J) * Price( J),'%6.2f'), ' - Cost: $',
      @FORMAT(PRODUCE(J) * usage(i, j),'%6.2f'), ' = Profit: $',
      @FORMAT(PRODUCE(J) * PRICE( j) - PRODUCE(J) * USAGE(I,J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX16);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/PRODUCT (un):
           PROD1     PROD2     PROD3
 INST1  8.000000  4.000000  10.00000
 INST2  9.000000  6.000000  10.00000
 INST3  12.00000  10.00000  11.00000

AVAILABLE (un):
 INST1  800.0000
 INST2  1000.000
 INST3  1200.000

DEMAND (un):
 PROD1  1000.000
 PROD2  900.0000
 PROD3  800.0000

PRICE (p/un):
 PROD1  20.00000
 PROD2  25.00000
 PROD3  30.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                                     47200.00
Infeasibilities:                                                                     0.000000

OPTIMAL PRODUCTION PROGRAM:
INST1  PROD3 Produce: 800 un x Price: $30.00 = Total: $24000.00 - Cost: $8000.00 = Profit: $16000.00
INST2  PROD3 Produce: 800 un x Price: $30.00 = Total: $24000.00 - Cost: $8000.00 = Profit: $16000.00
INST3  PROD3 Produce: 800 un x Price: $30.00 = Total: $24000.00 - Cost: $8800.00 = Profit: $15200.00
```

17

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities
- Purchase

Source:

- Book 2
- Page 231

GOAL

One company manufactures two types of heater: one electric and one gas model and has just signed a contract to provide 30,000 electric heaters and 15,000 gas.

However, its productive capacity is limited to three processes, in the number of hours available.

Alternatively, the company can buy similar heaters on the market, paying for the $67 electric model and the $95 gas model.

All the necessary information for the development of the model follows below:

Resources / Products			Electric	Gas	Available
Process	Production	hr	0.2	0.4	10,000
	Assembler	hr	0.3	0.1	15,000
	Package	hr	0.1	0.1	5,000
Demand		un	30,000	15,000	-
Production Cost		$	55.00	85.00	-
Purchase Cost		$	67.00	95.00	-

Determine how many heaters of each model are to be manufactured and how many must be purchased in order to meet the request at the lowest possible cost.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE, DEMAND;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Available resources;
 RESOURCE,        AVAILABLE    =
 PRODUCTION       10000
 ASSEMBLER        15000
 PACKAGE          5000;

! Product Attributes;
 PRODUCT,         COST,        DEMAND  =
 ELE_PRODU        55           30000
 GAS_PRODU        85           15000
 ELE_PURCH        67           0
 GAS_PURCH        95           0;

! Required p/unit (hr)  ELE_PRODU  GAS_PRODU  ELE_PURCH  GAS_PURCH;
 USAGE  =               0.2         0.4        0          0                ! PRODUCTION;
                        0.3         0.1        0          0                ! ASSEMBLER;
                        0.1         0.1        0          0 ;              ! PACKAGE;
ENDDATA
SUBMODEL MIN17:
MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [CAP] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
! The Demand constraints;
PRODUCE( 1) + PRODUCE( 3) = 30000;
PRODUCE( 2) + PRODUCE( 4) = 15000;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/PRODUCT (hour):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (hour):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  PRODUCTION/PURCHASE  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN17);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION/PURCHASE PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '    ',
        @FORMAT(PRODUCT(J),'-10s'), @IF(@STRLEN(PRODUCT(J)) #GT# 11, 'Purchase:' ,'Produce: ' ),
        @FORMAT(PRODUCE( J),'%5.0f'),' Heater x  ', 'Unit cost: $',
        @FORMAT(COST( J),'%5.2f'),'  =  ', 'Total: $',
        @FORMAT(COST( J) * PRODUCE(J),'%10.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN17);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/PRODUCT (hour):
             ELE_PRODU  GAS_PRODU  ELE_PURCH  GAS_PURCH
 PRODUCTION  0.2000000  0.4000000   0.000000   0.000000
 ASSEMBLER   0.3000000  0.1000000   0.000000   0.000000
 PACKAGE     0.1000000  0.1000000   0.000000   0.000000

AVAILABLE (hour):
 PRODUCTION  10000.00
 ASSEMBLER   15000.00
 PACKAGE     5000.000

DEMAND (un):
 ELE_PRODU   30000.00
 GAS_PRODU   15000.00
 ELE_PURCH   0.000000
 GAS_PURCH   0.000000

PRODUCTION/PURCHASE COST:
 ELE_PRODU   55.00000
 GAS_PRODU   85.00000
 ELE_PURCH   67.00000
 GAS_PURCH   95.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                2975000.
Infeasibilities:                                                0.000000

OPTIMAL PRODUCTION/PURCHASE PROGRAM:
  ELE_PRODU Produce: 30000 Heater x  Unit cost: $55.00  =  Total: $1650000.00
  GAS_PRODU Produce: 10000 Heater x  Unit cost: $85.00  =  Total: $ 850000.00
  GAS_PURCH Produce:  5000 Heater x  Unit cost: $95.00  =  Total: $ 475000.00
```

18

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Blend

Source:

- Book 2
- Page 233

GOAL

An agro-industrial company produces 3 types of canning. Each type requires a similar industrial treatment that differs in duration.

- 1000 boxes of tomato soup requires 200 hours of labor + 6 hours of equipment
- 1000 boxes of tomato juice requires 80 hours of labor + 24 hours of equipment
- 1000 boxes of tomato sauce requires 300 hours of labor + 7 hours of equipment.

All the necessary information for the development of the model follows below:

Resources / Products		Soup	Juice	Sauce	Available
Labor	hr	200	80	300	5,000
Equipment	hr	6	24	7	168
Minimum Production	pac	500	500	500	-
Profit	$	3.00	2.50	1.00	-

Assuming that the company wants to maximize its profit, determine the production of each of the products,

```
MODEL:
SETS:
 PRODUCT : PROFIT, MINPROD, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Available resources;
 RESOURCE ,          AVAILABLE  =
 LABOR               5000
 EQUIPMENT           168;
! Product Attributes;
 PRODUCT,            PROFIT,    MINPROD  =
 SOUP                3          500
 JUICE               2.5        500
 SAUCE               1          500;

! Required p/unit (hr)  SOUP    JUICE     SAUCE;
 USAGE  =               200     80        300       ! Labor;
                        6       24        7;        ! Equipment;
ENDDATA
SUBMODEL MAX18:
MAX =  @SUM( PRODUCT (p): PRODUCE( p) * PROFIT( p));
! The capacity constraints;
@FOR( RESOURCE( I):
        [CAP] @SUM( PRODUCT(J): USAGE(I, J)/1000 * PRODUCE(J)) <= AVAILABLE( I););
! The demand constraints;
@FOR( PRODUCT(K):
        [DEM] @SUM( PRODUCT( K): PRODUCE( K )) >= MINPROD( K));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/PRODUCT (hr):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (hr):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (Pac):", @NEWLINE( 1));
@TABLE(MINPROD);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/pac):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX18);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
        @FORMAT(PRODUCT(J),'-5s'),' Produce:',
        @FORMAT(PRODUCE( J),'%5.0f'), ' Pac  x  ', 'Unit profit: $',
        @FORMAT(PROFIT( J),'%4.2f'), ' = ', 'Total: $',
        @FORMAT(PRODUCE(J) * Profit( J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX18);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/PRODUCT (hr):
                SOUP     JUICE     SAUCE
 LABOR      200.0000  80.00000  300.0000
 EQUIPMENT  6.000000  24.00000  7.000000

AVAILABLE (hr):
 LABOR      5000.000
 EQUIPMENT  168.0000

MINIMUM PRODUCTION (Pac):
 SOUP       500.0000
 JUICE      500.0000
 SAUCE      500.0000

PROFIT (p/pac):
 SOUP       3.000000
 JUICE      2.500000
 SAUCE      1.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                         74393.52
Infeasibilities:                                         0.000000

OPTIMAL PRODUCTION PROGRAM:
SOUP  Produce:23898 Pac  x  Unit profit: $3.00 = Total: $71694.44
JUICE Produce:  880 Pac  x  Unit profit: $2.50 = Total: $ 2199.07
SAUCE Produce:  500 Pac  x  Unit profit: $1.00 = Total: $  500.00
```

19

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

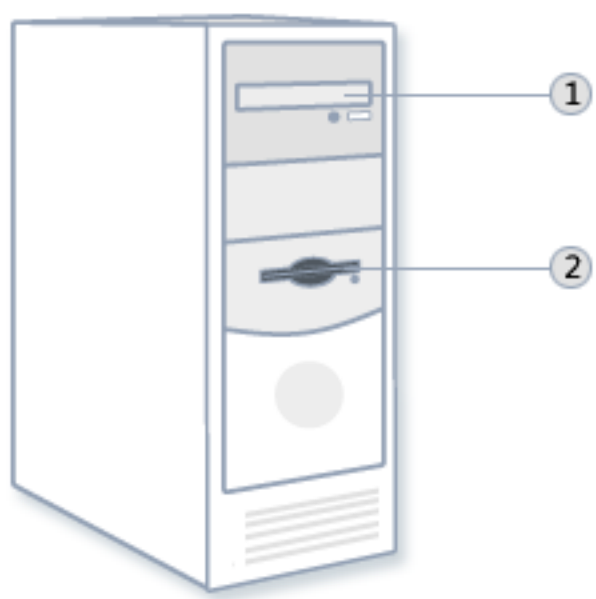
Source:

- Book 4

GOAL

As a manufacturer of electronic equipment, you must design a cabinet for your new product, which meets the area, volume, marketing and aesthetic requirements.

In this example, we create a nonlinear optimization model to design the case for a computer.



```
MODEL:
! Design a box at minimum cost that meetsarea, volume, marketing and aesthetic requirements;
SUBMODEL MIN19:
 [COST] MIN = 2*( .05*(d*w + d*h) +.1*w*h);
 [SURFACE]  2*(h*d + h*w + d*w) >= 888;
 [VOLUME]   h*d*w >= 1512;
! These two enforce aesthetics;
 [NOTNARRO] h/w <= .718;
 [NOTHIGH]  h/w >= .518;
! Marketing requires a small footprint;
 [FOOTPRNT] d*w <= 252;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
!Data block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("    VOLUME:    1512", @NEWLINE( 1));
@WRITE("    SURFACE:   888", @NEWLINE( 1));
@WRITE("    NOTNARRO:  0.718", @NEWLINE( 1));
@WRITE("    NOTHIGH:   0.518", @NEWLINE( 1));
@WRITE("    FOOTPRNT:  252", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN19);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  DIMENSIONS OF THE BOX: ", @NEWLINE( 1));
@WRITE( '    Depth:   ',
@FORMAT(D,'%8.5f'), @NEWLINE( 1), '    Width:   ',
@FORMAT(W,'%8.5f'), @NEWLINE( 1),'    Height:  ',
@FORMAT(H,'%8.5f'), @NEWLINE( 2), '  MINIMUM COST:', 32*' ','$',
@FORMAT(COST, '%8.5F'), @NEWLINE( 1));
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

  VOLUME:    1512
  SURFACE:   888
  NOTNARRO:  0.718
  NOTHIGH:   0.518
  FOOTPRNT:  252
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Local optimal solution found.
Objective value:                              50.96508
Infeasibilities:                              0.000000
Extended solver steps:                               5
Best multistart solution found at step:              1
Total solver iterations:                            51
Elapsed runtime seconds:                          0.54


DIMENSIONS OF THE BOX:
  Depth:   23.03096
  Width:    9.56220
  Height:   6.86566

MINIMUM COST:                                $50.96508
```

20

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 233

GOAL

One company manufactures some models of garden equipment. The company intends to determine how many units to produce during the next few months, in order to meet demand at the lowest possible cost.

The company estimates a cost of $ 1.50 per month for each unit held in stock.

It considers that the number of units in stock per month is the average of the quantity existing at the beginning of the month. Currently it has 120 units in stock.

To maintain the workforce, it needs to produce at least 400 units per month and also wants to keep a minimum safety stock of 50 units per month.

The company intends to determine how many units it will produce during the next 4 months, in order to meet the demand for the lowest possible cost.

All the necessary information for the development of the model follows below:

Items / Month		M1	M2	M3	M4
Production Cost	$	49.00	45.00	46.00	47.00
Cost of Stock	$	1.50	1.50	1.50	1.50
Initial and Safety Stock	un	120	50	50	50
Demand	un	420	580	310	540
Minimum Production	un	400	400	400	400
Production Capacity	un	500	520	450	550

```
MODEL:
SETS:
 PRODUCT:  COST, COST_STOCK, USED_STOCK, STK_BAL,  MINPROD, DEMAND, CAPACITY, PRODUCE;
 HEADER  / PROD, LIMIT, VALUE /:;
ENDSETS
DATA:
! Product Attributes;
  PRODUCT,   COST,        COST_STOCK,       MINPROD,     DEMAND,      CAPACITY =
  M1         49.00        1.50              400          420          500
  M2         45.00        1.50              400          580          520
  M3         46.00        1.50              400          310          450
  M4         47.00        1.50              400          540          550;
ENDDATA
SUBMODEL MIN20:
 [OBJ] MIN =    @SUM(PRODUCT( p): COST( p) * PRODUCE( p) ) +
                @SUM(PRODUCT( p): STK_BAL( p) * COST_STOCK( p));
! Initial stock - Security stock;
USED_STOCK(1) = 120 - 110;
USED_STOCK(2) = 110 - 50;
USED_STOCK(3) = 140 - 50;
USED_STOCK(4) = 140 - 50;
! Stock balance;
STK_BAL(1) = 120 - 10;
STK_BAL(2) = 110 - 60;
STK_BAL(3) = 50  + 90;
STK_BAL(4) = 140 - 50;
! Equalization of production to meet demand;
PRODUCE(1) = DEMAND(1) - USED_STOCK(1);
PRODUCE(2) = DEMAND(2) - USED_STOCK(2);
! MinProd 400 un - Demand: 310 = Stock 90 + Security stock 50 un = 140un;
PRODUCE(3) = DEMAND(3) + USED_STOCK(3);
PRODUCE(4) = DEMAND(4) - USED_STOCK(4);
! minimum and maximum production range;
@FOR( PRODUCT(K):
        [MPR] @SUM( PRODUCT( K): PRODUCE( K)) >= MINPROD( K);
        [CAP]  @SUM( PRODUCT( K): PRODUCE( K)) <= CAPACITY( K););
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1), "  PRODUCTION COST p/un:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  COST OF STOCK (un):", @NEWLINE( 1));
@TABLE(COST_STOCK);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  MAXIMUM PRODUCTION (un):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (un):", @NEWLINE( 1));
@TABLE(MINPROD);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN20);
! Solution report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 2));
@WRITE( '  PRODUCTION COST:', @NEWLINE(1));
@WRITEFOR( PRODUCT( J)| PRODUCE( J) #GT# 0: '  ',
@FORMAT(PRODUCT(J),'-6s'),'  Produce:',
@FORMAT(PRODUCE( J),'%4.0f'), ' un  x ', 'Unit cost: $',
@FORMAT(COST( J),'%5.2f'), ' = ', 'Total: $',
@FORMAT(PRODUCE(J) * COST( J),'%8.2f'),@NEWLINE( 1));
@WRITE('  TOTAL PRODUCTION COST:', 32*' ', '$',
@FORMAT( @SUM(PRODUCT( p): COST( p) * PRODUCE( p)),'%8.2f'), @NEWLINE(2));
! Cost of stock;
@WRITEFOR( PRODUCT( J)| PRODUCE( J) #GT# 0: '  ',
        @FORMAT(PRODUCT(J),'-6s'),'  Stock: ',
        @FORMAT(STK_BAL(J),'%5.0f'), ' un  x Cost:      $',
        @FORMAT(COST_STOCK(J),'%5.2f'), ' = Total: $',
        @FORMAT(STK_BAL * COST_STOCK(J),'%8.2f'), @NEWLINE(1));
        @WRITE('  COST OF STOCK:', 40*' ', '$',
        @FORMAT( @SUM(PRODUCT( J): STK_BAL( J) * COST_STOCK( J)),'%8.2f'),
@NEWLINE(2));
! Minimum Total Cost:;
@WRITE('  MINIMUM TOTAL COST::', 34*' ', '$', @FORMAT( OBJ,'%8.2f'), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN20);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

PRODUCTION COST p/un:     MAXIMUM PRODUCTION (un):
 M1  49.00000              M1  500.0000
 M2  45.00000              M2  520.0000
 M3  46.00000              M3  450.0000
 M4  47.00000              M4  550.0000

COST OF STOCK (un):       MINIMUM PRODUCTION (un):
 M1  1.500000              M1  400.0000
 M2  1.500000              M2  400.0000
 M3  1.500000              M3  400.0000
 M4  1.500000              M4  400.0000

DEMAND (un):
 M1  420.0000
 M2  580.0000
 M3  310.0000
 M4  540.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                       83625.00
Infeasibilities:                                       0.000000

OPTIMAL PRODUCTION PROGRAM:

PRODUCTION COST:
M1      Produce: 410 un  x Unit cost: $49.00 = Total: $20090.00
M2      Produce: 520 un  x Unit cost: $45.00 = Total: $23400.00
M3      Produce: 400 un  x Unit cost: $46.00 = Total: $18400.00
M4      Produce: 450 un  x Unit cost: $47.00 = Total: $21150.00
TOTAL PRODUCTION COST:                                $83040.00

M1      Stock:   110 un  x Cost:      $ 1.50 = Total: $  165.00
M2      Stock:    50 un  x Cost:      $ 1.50 = Total: $   75.00
M3      Stock:   140 un  x Cost:      $ 1.50 = Total: $  210.00
M4      Stock:    90 un  x Cost:      $ 1.50 = Total: $  135.00
COST OF STOCK:                                        $  585.00

MINIMUM TOTAL COST:                                   $83625.00
```

21

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Factory

Source:
- Book 2
- Page 235

GOAL

A wine producer has 2 vineyards whose production capacity and cost per bottle are detailed below.

Four Italian restaurants located near the wineries are interested in buying this wine.

The demand required for each restaurant, as well as price, shipping costs, are also detailed below:

Winery / Restaurants			ST1	ST2	ST3	ST4	Capacity	Cost p/un
Demand		un	1800	2300	1250	1750	-	-
Shipping	W1	$	7.00	8.00	13.00	9.00	3500 un	$23.00
	W2	$	12.00	6.00	8.00	7.00	3100 un	$25.00
Price		$	69.00	67.00	70.00	66.00	-	-

The producer wants to determine the production and the shipping plan in order to maximize the profits

```
MODEL:
SETS:
WINERY: CAPACITY , COST;
VENDORS: DEMAND,PRICE;
LINKS( WINERY, VENDORS): SHIPPING, VOLUME;
ENDSETS
DATA:
! Winery attributes;
WINERY,            CAPACITY,    COST   =
WINERY1            3500         23.00
WINERY2            3100         25.00;
! Vendor attributes;
VENDORS,           DEMAND,      PRICE          =
ST1                1800         69.00
ST2                2300         67.00
ST3                1250         70.00
ST4                1750         66.00;
! Shipping p/bottle ST1     ST2    ST3     ST4;
SHIPPING   =       7       8      13      9                 ! WINERY1;
                   12      6      8       7;                ! WINERY2;
ENDDATA
SUBMODEL MAX21:
[OBJ] MAX = @SUM( LINKS( I, J): PRICE( J) * VOLUME( I,J) - SHIPPING( I, J) * VOLUME( I, J) - COST(I) * VOLUME( I,J));
! The demand constraints;
@FOR( VENDORS( J):
      [DEM]  @SUM( WINERY( I): VOLUME( I, J)) <= DEMAND( J));
! The capacity constraints;
@FOR( WINERY( I):
      [CAP]   @SUM( VENDORS( J): VOLUME( I, J)) = CAPACITY( I));
ENDSUBMODEL
CALC:
@SET('TERSEO',1);   ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);    ! Post status windows, 1 Yes, 0 No;
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   SHIPPING COST:", @NEWLINE( 1));
@TABLE(SHIPPING);
@WRITE(" ", @NEWLINE( 1), " CAPACITY (bottle):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (bottle):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  BOTTLE COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX21);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( LINKS( I, J) | VOLUME( I,J) #GT# 0: '  ', WINERY(I),' To:', VENDORS(j),@NEWLINE(1), '      ',
      @FORMAT(VOLUME( I, J),'%4.0f'), ' bottle x ', 'price: $',
      @FORMAT(PRICE( J),'%8.2f'), ' = Revenue: $',
      @FORMAT(PRICE(J) * VOLUME(I, J),'%9.2f'), @NEWLINE(1),36*' ','- Cost:    $',
      @FORMAT(COST(I)  * VOLUME(I, J),'%9.2f'),  @NEWLINE(1),36*' ','- Ship:    $',
      @FORMAT(SHIPPING( I, J) * VOLUME(I,J),'%9.2f'), @NEWLINE(1),36*' ','= Profit:  $',
      @FORMAT(PRICE(J) * VOLUME( I, J) - COST(I) * VOLUME(I, J) - SHIPPING( I, J) * VOLUME(I,J),'%9.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITEFOR( LINKS( I, J) | VOLUME( I,J) #GT# 0: '  ', WINERY(I),' To:', VENDORS(j),20*' ','= Profit:  $',
      @FORMAT(PRICE(J) * VOLUME( I, J) - COST(I) * VOLUME(I, J) - SHIPPING( I, J) *VOLUME(I,J),'%9.2f'),
@NEWLINE(1));
@WRITE('  TOTAL PROFIT:', 32*' ', '$',
@FORMAT(@SUM(LINKS(I,J):PRICE(J) * VOLUME( I, J) - COST(I) * VOLUME(I, J) - SHIPPING( I, J) * VOLUME(I,J)),'%9.2f'), @
NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX21);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 SHIPPING COST:
              ST1       ST2       ST3       ST4
 WINERY1  7.000000  8.000000  13.00000  9.000000
 WINERY2  12.00000  6.000000  8.000000  7.000000

 CAPACITY (bottle):
   WINERY1   3500.000
   WINERY2   3100.000

 BOTTLE COST:
  WINERY1   23.00000
  WINERY2   25.00000

 DEMAND (bottle):
  ST1   1800.000
  ST2   2300.000
  ST3   1250.000
  ST4   1750.000

 PRICE:
  ST1   69.00000
  ST2   67.00000
  ST3   70.00000
  ST4   66.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              241750.0
Infeasibilities:                              0.000000

OPTIMAL PROGRAM:
WINERY1 To:ST1
   1800 bottle x price: $   69.00 = Revenue: $124200.00
                                  - Cost:    $ 41400.00
                                  - Ship:    $ 12600.00
                                  = Profit:  $ 70200.00
WINERY1 To:ST2
    450 bottle x price: $   67.00 = Revenue: $ 30150.00
                                  - Cost:    $ 10350.00
                                  - Ship:    $  3600.00
                                  = Profit:  $ 16200.00
WINERY1 To:ST4
   1250 bottle x price: $   66.00 = Revenue: $ 82500.00
                                  - Cost:    $ 28750.00
                                  - Ship:    $ 11250.00
                                  = Profit:  $ 42500.00
WINERY2 To:ST2
   1850 bottle x price: $   67.00 = Revenue: $123950.00
                                  - Cost:    $ 46250.00
                                  - Ship:    $ 11100.00
                                  = Profit:  $ 66600.00
WINERY2 To:ST3
   1250 bottle x price: $   70.00 = Revenue: $ 87500.00
                                  - Cost:    $ 31250.00
                                  - Ship:    $ 10000.00
                                  = Profit:  $ 46250.00

WINERY1 To:ST1                    = Profit:  $ 70200.00
WINERY1 To:ST2                    = Profit:  $ 16200.00
WINERY1 To:ST4                    = Profit:  $ 42500.00
WINERY2 To:ST2                    = Profit:  $ 66600.00
WINERY2 To:ST3                    = Profit:  $ 46250.00
TOTAL PROFIT:                                $241750.00
```

22

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Blend

Source:

- Book 3
- Chapter 2.3.8

GOAL

A juice bottler prepares Blueberry honey in two options: Dark and Light. Each type is constituted by a different proportion among the basic components.

Since water is abundant and does not have significant weight in the final price of the product, is not considered in the production planning of each barrel:

Following is all the information necessary to develop the model, considering Formula, Availability and Profit per barrel:

Resources / Products		Dark	Light	Available
Citrus	kg/barrel	5	15	480
Glucose	kg/barrel	4	4	160
Blueberry	kg/barrel	35	20	1190
Price p/barrel	$	10.00	25.00	-

Therefore, elaborate a program of production of these types of juice in order to maximize profit per barrel.

```
MODEL:
SETS:
  PRODUCT : PROFIT, PRODUCE;
  HEADER  / DARK, LIGHT, LIMIT, VALUE /:;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
  PXR( RESOURCE, HEADER)   :  SLASUR;
ENDSETS
DATA:
! Available resources;
  RESOURCE,                AVAILABLE =
  CITRUS_PECTIN            480
  GLUCOSE                  160
  BLUEBERRY                1190;
! Product attributes;
  PRODUCT,                 PROFIT =
  DARK                     10
  LIGHT                    25;
! Required p/barrel (kg)   DARK        LIGHT;
  USAGE      =             5           15           ! CITRUS_PECTIN;
                           4           4            ! GLUCOSE;
                           35          20;          ! BLUEBERRY;
ENDDATA
SUBMODEL MAX5:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (kg):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT per barrel:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX5);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE( J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J),'-6s'),
        @FORMAT(PRODUCE( J),'%2.0f'),' Barrel x Unit profit: $',
        @FORMAT(PROFIT( J),'%4.2f'), ' = Total: $',
        @FORMAT(PROFIT( J) * PRODUCE( J),'%6.2f'),
@NEWLINE( 1));
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  SLACK/SURPLUS LIMIT = AVAILABLE: ", @NEWLINE( 1));
@FOR(PXR(I,J): SLASUR(I,3) = AVAILABLE(I));
@FOR(RXP(I,J)| J #LT# 3: SLASUR(I,J) = PRODUCE(J) * USAGE(I,J));
@FOR(PXR(I,J): SLASUR(I,4) = SLASUR(I,1) + SLASUR(I,2)  -  SLASUR(I,3));
@TABLE(SLASUR);
! Execute the Graph;
@CHARTPIE( 'Product Mix Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (kg/barrel):
                    DARK     LIGHT
 CITRUS_PECTIN  5.000000  15.00000
 GLUCOSE        4.000000  4.000000
 BLUEBERRY      35.00000  20.00000

AVAILABLE (kg):
 CITRUS_PECTIN  480.0000
 GLUCOSE        160.0000
 BLUEBERRY      1190.000

PROFIT p/barrel:
 DARK           10.00000
 LIGHT          25.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                               820.0000
Infeasibilities:                               0.000000

IDEAL PRODUCTION PROGRAM:
DARK  12 Barrel x Unit profit: $10.00 = Total: $120.00
LIGHT 28 Barrel x Unit profit: $25.00 = Total: $700.00

SLACK/SURPLUS LIMIT = AVAILABLE:
                     DARK      LIGHT      LIMIT       VALUE
 CITRUS_PECTIN   60.00000   420.0000   480.0000    0.000000
 GLUCOSE         48.00000   112.0000   160.0000    0.000000
 BLUEBERRY       420.0000   560.0000   1190.000   -210.0000
```

23

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

GOAL

A confection operates with two products: trousers and shirts. Because they are similar products, they have comparable productivity and share the same features.

Production scheduling is performed by product batches. The production department says it will take 15 men an hour for a batch of pants and 20 men an hour for a batch of shirts. It is known that no specialized manpower is required for the production of fighters, but it takes 10 men an hour of this type of labor to produce a batch of shirts.

Skilled workers can produce shirts or trousers, and 30 men per hour are available from skilled workers and 50 non-skilled men per hour.

From the production plant it is known that there are only two machines capable of producing two types of product, and machine 1 can produce a batch of shirts every 10 hours, and can not be used for more than 80 hours in the period considered.

Machine 2 can produce a batch of pants every 30 hours and a batch of shirts every 35 hours, and can not be used for more than 130 hours in the period considered.

Two types of raw material are required to produce trousers and shirts. In the production of a batch of pants, 12 kilos of raw material A, 10 kilos of B.

In the production of a batch of shirts, 8 kilos of raw material A and 15 kilos of B.

The warehouse informs that, by imposing space, it can only to provide 120 kg of A and 100 kg of B in the period considered.

Knowing that price by sale is $800 in lots of shirts and $500 in lots of pants. All the necessary information for the development of the model follows below:

Resources / Products			Shirts	Pants	Available
Worker	T1N	head	20	15	80
	T1E	head	10	0	30
Machine	1	hr	10	20	80
	2	hr	35	30	130
Raw	A	k	8	12	120
	B	k	15	10	100
Price (p/lot)		$	800.00	500.00	-

Formulate the problem in PL to maximize revenue.

```
MODEL:
SETS:
  PRODUCT : PRICE, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Available resources;
  RESOURCE,             AVAILABLE =
  T1N_WORKER            80
  T1E_WORKER            30
  MACHINE1              80
  MACHINE2              130
  RM_A                  120
  RM_B                  100;
! Product attributes;
  PRODUCT,              PRICE =
  SHIRTS                800.00
  PANTS                 500.00;

! Required              SHIRTS      PANTS;
  USAGE =               20          15          ! T1_WORKER (head);
                        10          0           ! T2_WORKER (head);
                        10          20          ! MACH1NE1 (hr);
                        35          30          ! MACH1NE2 (hr);
                        8           12          ! RM_A (k);
                        15          10;         ! RM_B (k);
ENDDATA
SUBMODEL MAX23:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "   RESOURCES/USAGE ( head, head, hr, hr, k, k):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE ( head, head, hr, hr, k, k):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX23);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
      @FORMAT(PRODUCT(J),'-7s'), ' produce:',
      @FORMAT(PRODUCE( J),'%2.1f'),' lots x price: $',
      @FORMAT(PRICE( J),'%6.2f'),'  = $',
      @FORMAT(PRICE(J) * PRODUCE(J),'%7.2f'),
@NEWLINE( 1));
@CHARTPIE( 'Product Mixer Model', 'Produce', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX23);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE ( head, head, hr, hr, k, k):
              SHIRTS     PANTS
 T1N_WORKER  20.00000  15.00000
 T1E_WORKER  10.00000  0.000000
 MACHINE1    10.00000  20.00000
 MACHINE2    35.00000  30.00000
 RM_A        8.000000  12.00000
 RM_B        15.00000  10.00000

AVAILABLE ( head, head, hr, hr, k, k):
 T1N_WORKER  80.00000
 T1E_WORKER  30.00000
 MACHINE1    80.00000
 MACHINE2    130.0000
 RM_A        120.0000
 RM_B        100.0000

PRICE:
 SHIRTS  800.0000
 PANTS   500.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              2816.667
Infeasibilities:                              0.000000

OPTIMAL PRODUCTION PROGRAM:
SHIRTS  produce:3.0 lots x price: $800.00  =  $2400.00
PANTS   produce:0.8 lots x price: $500.00  =  $ 416.67
```

24

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 237

GOAL

A company produces only three items: Rollerball pen, mechanical pencils and fountain pen.

The unitary income for each of the items is $3.00 for rollerball pen $3.00 to $5.00 and mechanical pencil with fountain pen.

The company is planning its production mix for next week. It is believed that the company can sell any amount of pens and mechanical pencils to produce, but the production is limited to available resources.

All the necessary information for the development of the model follows below:

Resources / Products		Rollerball Pen	Automatic Pencil	Fountain Pen	Available
Plastic	g	1.2	1.7	1.2	1,000
Chrome	g	0.8	0	2.3	1,200
Stainless Steel	g	2.0	3.0	4.5	2,000
Price	$	3.00	3.00	5.00	-

Based on this information, build a linear programming model that allows the company to plan next week's production for maximum revenue.

```
MODEL:
SETS:
 PRODUCT : PRICE, PRODUCE;
 HEADER  / ROLPEN, AUTPEN, FOUPEN, LIMIT, VALUE /:;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
 PXR( RESOURCE, HEADER)   :  SLASUR;
ENDSETS
DATA:
! Resources  attributes;
 RESOURCE ,               AVAILABLE    =
 PLASTIC                  1000
 CHROME                   1200
 ST_STEEL                 2000;
! Product attributes;
 PRODUCT,                 PRICE   =
 ROLPEN                   3.00
 AUTPEN                   3.00
 FOUPEN                   5.00;
! Required (g)            ROLPEN       AUTPEN       FOUPEN;
 USAGE  =                 1.2          1.7          1.2          !  PLASTIC ;
                          0.8          0            2.3          !  CHROME ;
                          2.0          3.0          4.5;         !  ST_STEEL;
ENDDATA
SUBMODEL MAX24:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
@SET('TERSEO',1);    ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);     ! Post status windows, 1 Yes, 0 No;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE("   RESOURCES/USAGE (g):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1));
@WRITE("  AVAILABLE (g):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1));
@WRITE("  PRICE :", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1));
@WRITE("  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX24);
@WRITE("  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
         @FORMAT(PRODUCT(J),'-6s'),': Produce:',
         @FORMAT(PRODUCE( J),'%3.0f'),' un x price: $',
         @FORMAT(PRICE( J),'%4.2f'),' = ', 'Total: $',
         @FORMAT(PRICE(J) * PRODUCE(J),'%7.2f'),
@NEWLINE( 1));
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  SLACK/SURPLUS LIMIT = AVAILABLE: ", @NEWLINE( 1));
@FOR(PXR(I,J): SLASUR(I,4) = AVAILABLE(I));
@FOR(RXP(I,J)| J #LT# 4: SLASUR(I,J) = PRODUCE(J) * USAGE(I,J));
@FOR(PXR(I,J): SLASUR(I,5) = SLASUR(I,1) + SLASUR(I,2) + SLASUR(I,3) -  SLASUR(I,4));
@TABLE(SLASUR);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX24);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RESOURCES/USAGE (g):
              ROLPEN     AUTPEN     FOUPEN
 PLASTIC     1.200000   1.700000   1.200000
 CHROME     0.8000000   0.000000   2.300000
 ST_STEEL    2.000000   3.000000   4.500000

AVAILABLE (g):
 PLASTIC    1000.000
 CHROME     1200.000
 ST_STEEL   2000.000

PRICE :
 ROLPEN  3.000000
 AUTPEN  3.000000
 FOUPEN  5.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              2766.667
Infeasibilities:                              0.000000

OPTIMAL PRODUCTION PROGRAM:
ROLPEN: Produce:700 un x price: $3.00 = Total: $2100.00
FOUPEN: Produce:133 un x price: $5.00 = Total: $ 666.67

SLACK/SURPLUS LIMIT = AVAILABLE:
              ROLPEN     AUTPEN     FOUPEN      LIMIT      VALUE
 PLASTIC    840.0000   0.000000   160.0000   1000.000   0.000000
 CHROME     560.0000   0.000000   306.6667   1200.000  -333.3333
 ST_STEEL   1400.000   0.000000   600.0000   2000.000   0.000000
```

25

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory

Source:

- Book 2
- Page 240

GOAL

A paper recycling company transforms paper, sulfite paper, carbon and several other types of paper into a pulp or dough from which paper, paperboard and butter paper can be produced.

The following table summarizes the percentage of recovered paper obtained from the papers that are recycled.

Formulas		Newspaper	Cardboard	Parchment	Availability
Newspaper	%	85	80	0	600
Various pulp	%	90	80	70	500
Sulfite Paper	%	90	85	80	300
Carbon Paper	%	80	70	0	400
Target Production	Ton	500	600	300	-
Processing Cost		Newspaper	Cardboard	Parchment	Purchase
Newspaper	$	6.50	11.00	0.00	15.00
Various pulp	$	9.75	12.25	9.50	16.00
Sulfite Paper	$	4.75	7.75	8.50	19.00
Carbon Paper	$	7.50	8.50	0.00	17.00

To better understand the previous table, if we have 1 ton of newspaper we can recover 0.85 ton of newsprint or 0.80 ton of cardboard. If we have 1 ton of carbon paper to be recycled we can get 0.80 ton of newsprint or 0.70 ton of cardboard.

The costs for processing each ton of material (papers) in the various types of pulp and respective procurement costs are also indicated in the table above.

The company wants to determine the highest production of pulp at the lowest cost, to produce 500 tons of paper pulp, 600 tons of pulp and 300 tons of paper.

```
MODEL:
SETS:
 PRODUCT : TARGET, PRODUCE;
 RESOURCE: AVAILABLE, PURCHASE, CP, PC;
 RXP( RESOURCE, PRODUCT): FORMULA, COST_PROC, BUILD;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,      AVAILABLE,        PURCHASE   =
 NEWSPAPER      600               15.00
 VARIOUS        500               16.00
 SULFIT         300               19.00
 CARBON         400               17.00;

! Product attributes;
 PRODUCT,       TARGET      =
 NEWSPAPER      500
 CARDBOARD      600
 PARCHMENT      300;

! Required      Newspaper   Cardboard   Parchment;
 FORMULA    =   85          80          0           ! Paper to be recycled: Newspaper;
                90          80          70          ! Paper to be recycled: Various paper;
                90          85          80          ! Paper to be recycled: Sulfite paper;
                80          70          0;          ! Paper to be recycled: Carbon paper;

 COST_PROC  =   6.50        11.00       0.00        ! Processing cost: Newspaper;
                9.75        12.25       9.50        ! Processing cost: Various paper;
                4.75        7.75        8.50        ! Processing cost: Sulfite paper;
                7.50        8.50        0.00;       ! Processing cost: Carbon paper;

! Select              NEWSPAPER  CARDBOARD PARCHMENT;
 BUILD      =   1           0           0           ! Or;
                0           0           1           ! Or;
                0           1           0           ! Or;
                0           1           0;          ! Or;
ENDDATA
SUBMODEL MAX25:
[OBJ] MAX = @SUM(PRODUCT(J): PRODUCE(J));
! Purchase cost;
@FOR(RESOURCE(I):
      CP(I) = @SUM( PRODUCT(J): PRODUCE(J) * (1+(1-FORMULA(I,J)/100)) * PURCHASE(I) * BUILD(I,J)););
! Processing cost;
@FOR(RESOURCE(I):
      PC(I) =  @SUM( PRODUCT(J): PRODUCE(J) * (1+(1-FORMULA(I,J)/100)) * COST_PROC(I,J) * BUILD(I,J)));
! The available constraints;
@FOR(RESOURCE(I):
      @SUM( PRODUCT(J): PRODUCE(J) * (1+(1-FORMULA(I,J)/100)) * BUILD(I,J)) <= AVAILABLE(I););
! The Target constraints;
@FOR(PRODUCT(J):
      @SUM( RESOURCE(I): PRODUCE(J) ) >= TARGET(J););
@FOR( RXP(I,J): @BIN(BUILD));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   RECYCLING FORMULA (%):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "   PROCESSING COST P/TON:", @NEWLINE( 1));
@TABLE(COST_PROC);
@WRITE(" ", @NEWLINE( 1), "  PURCHASE PROFIT P/TON:", @NEWLINE( 1));
@TABLE(PURCHASE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Ton):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  TARGET (Ton): ", @NEWLINE( 1));
@TABLE(TARGET);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX25);
! Solution report;
@WRITE(' ',@NEWLINE( 1), '   PRODUCE:',@NEWLINE( 1));
@WRITEFOR(PRODUCT(I): '   . ',
        @FORMAT(PRODUCT(I),'-10s'), 31*' ',
        @FORMAT(PRODUCE(I),'%7.1f'), ' ton',
@NEWLINE(1));
@WRITE(' ',@NEWLINE( 1));
! Processing cost;
@WRITE('  PROCESSING COST:',@NEWLINE( 1));
@WRITEFOR( RXP( I, J)  | BUILD(I,J) #GT# 0: '   . ',
        @FORMAT(RESOURCE(I),'-10S'),
        @FORMAT(PRODUCE(J) * (1+(1-FORMULA(I,J)/100)) * BUILD(I,J), '%7.1f'),' ton  x  cost: $',
        @FORMAT(COST_PROC(I,J),'%5.2f'),'  =  $',
        @FORMAT(PRODUCE(J) * COST_PROC(I,J),'%8.2f'),
@NEWLINE( 1));
@WRITE( '  Total:', 40*' ', '$',@FORMAT(@SUM(PRODUCT(J): PC(J)),'%8.2f'),@NEWLINE(1));
! Purchase cost;
@WRITE(" ", @NEWLINE( 1), '  PURCHASE COST:',@NEWLINE( 1));
@WRITEFOR( RXP( I, J)  | BUILD(I,J) #GT# 0: '   . ',
        @FORMAT(RESOURCE(I),'-10S'),
        @FORMAT(PRODUCE(J) * (1+(1-FORMULA(I,J)/100)) * BUILD(I,J), '%7.1f'),' ton  x  cost: $',
        @FORMAT(PURCHASE(I),'%5.2f'),'  =  $',
        @FORMAT(PRODUCE(J) * PURCHASE(I),'%8.2f'),
@NEWLINE( 1));
@WRITE( '  Total:', 40*' ', '$',@FORMAT(@SUM(PRODUCT(J): CP(J)),'%8.2f'),@NEWLINE(2));
! Total cost ( Processing + Purchase );
@WRITE('  TOTAL COST (PROCESSING + PURCHASE):', 11*' ', '$',
@FORMAT(@SUM(PRODUCT(J): CP(J)+PC(J)),'%8.2f'),@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX25);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 RECYCLING FORMULA (%):
            NEWSPAPER  CARDBOARD  PARCHMENT
 NEWSPAPER   85.00000   80.00000   0.000000
 VARIOUS     90.00000   80.00000   70.00000
 SULFIT      90.00000   85.00000   80.00000
 CARBON      80.00000   70.00000   0.000000

 PROCESSING COST P/TON:
            NEWSPAPER  CARDBOARD  PARCHMENT
 NEWSPAPER   6.500000   11.00000   0.000000
 VARIOUS     9.750000   12.25000   9.500000
 SULFIT      4.750000   7.750000   8.500000
 CARBON      7.500000   8.500000   0.000000

 PURCHASE COST P/TON:
  NEWSPAPER   15.00000
  VARIOUS     16.00000
  SULFIT      19.00000
  CARBON      17.00000

 AVAILABLE (Ton):
  NEWSPAPER   600.0000
  VARIOUS     500.0000
  SULFIT      300.0000
  CARBON      400.0000

 TARGET (Ton):
  NEWSPAPER   500.0000
  CARDBOARD   600.0000
  PARCHMENT   300.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              1167.224
Infeasibilities:                              0.000000

PRODUCE:
 . NEWSPAPER                                 521.7 ton
 . CARDBOARD                                 260.9 ton
 . PARCHMENT                                 384.6 ton

PROCESSING COST:
 . NEWSPAPER   600.0 ton  x  cost: $ 6.50  =  $ 3391.30
 . VARIOUS     500.0 ton  x  cost: $ 9.50  =  $ 3653.85
 . SULFIT      300.0 ton  x  cost: $ 7.75  =  $ 2021.74
 . CARBON      339.1 ton  x  cost: $ 8.50  =  $ 2217.39
Total:                                        $10975.00

PURCHASE COST:
 . NEWSPAPER   600.0 ton  x  cost: $15.00  =  $ 7826.09
```

26

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Facilities
- Team

Source:

- Book 3
- Chapter 2.15.13

GOAL

A company produces four different types of containers (C1, C2, C3 and C4). The profit per unit of each container produced and the weekly maximum demand of the containers in the market are described in the table below.

Resources / Products		C1	C2	C3	C4	Available
Welding	m/h	1	2	4	3	165
Installation / Door	m/h	3	4	2	3	75
Finish / Painter	m/h	2	5	1	7	210
Demand	un	45	65	90	65	-
Profit	$	10.00	15.00	18.00	21.00	-

Containers are assembled in three sequential processes. The critical aspect of the processes is the labor force.

The consumption of labor in each process and the quantity of men x hours are also described in the table above.

It is possible a scheme of reutilization of labor between the processes.

Consequently, it is possible that up to 15% of the workers who took part in the welding process are lent to the installation/door process.

Similarly, up to 20% of finishing and painting workers can work on installation /door.

Production requirements require that the ratio between containers C1 and C3 be between 0.8 and 1.1.

Formulate the problem to optimize the production of containers.

```
MODEL:
SETS:
PRODUCTS: DEMAND, PROFIT, VOLUME;
RESOURCE: AVAILABLE;
ROUTES( RESOURCE, PRODUCTS): USAGE ;
ENDSETS
DATA:
! Resource attributes;
RESOURCE,          AVAILABLE    =
WELDING            165
INST_DOOR          75
FINISH_PAINTER     210;
! Products attributes;
PRODUCTS,          DEMAND,            PROFIT  =
CONTAINER1         45                 10
CONTAINER2         65                 15
CONTAINER3         90                 18
CONTAINER4         65                 21;
!Required          Container1         Container2        Container3        Container4 ;
USAGE  =           1                  2                 4                 3              ! WELDING;
                   3                  4                 2                 3              ! INST_DOOR;
                   2                  5                 1                 7;             ! FINISH_PAINTER;
ENDDATA
SUBMODEL MAX26:
[OBJ] MAX = @SUM( PRODUCTS( J): PROFIT( J) * VOLUME(J));
! The demand constraints;
@FOR( PRODUCTS( J):
        [DEM] @SUM( products( J):  VOLUME( J)) <= DEMAND( J) * BUILD;
        @BIN(BUILD));
! The capacity constraints;
@FOR( RESOURCE( J):
        [AVA] @SUM( PRODUCTS( K): USAGE(J, K) * VOLUME( K)) <=  AVAILABLE( J););
! Proportion for transport ;
VOLUME(1) >= 0.8 * VOLUME(3);
VOLUME(1) <= 1.1 * VOLUME(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Sets the length of the line;
@SET('LINLEN',120);
! scheme of reutilization of labor between the processes;
! Installation / Door;
AVAILABLE(2) = AVAILABLE(2) + AVAILABLE(1) * 0.20 + AVAILABLE(3) * 0.15;
! Welding;
AVAILABLE(1) = AVAILABLE(1) * 0.80;
! Finish / painter;
AVAILABLE(3) = AVAILABLE(3) * 0.85;
! Data block;
@WRITE(" ", @NEWLINE( 1), "  REQUIRED ( m/hr ):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE ( m/hr ):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND ( m/hr):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX26);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITE('   PRODUCE:', @NEWLINE(1));
@WRITEFOR( PRODUCTS( J) | VOLUME( J) #GT# 0:'  ', PRODUCTS( J), ' produce:',
        @FORMAT(VOLUME( J),'%2.0f'), 'un x Unit profit: $',
        @FORMAT(PROFIT(J),'%4.2f'), ' = Total: $',
        @FORMAT(PROFIT(J) * VOLUME(J),'%6.2f'),
@NEWLINE( 1));
@WRITE('   Total:',48*' ','$', @FORMAT(OBJ,'%6.2f'), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX26);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED (m/hr):
                      C1        C2        C3        C4
 WELDING         1.000000  2.000000  4.000000  3.000000
 INST_DOOR       3.000000  4.000000  2.000000  3.000000
 FINISH_PAINTER  2.000000  5.000000  1.000000  7.000000

PROFIT:
 C1              10.00000
 C2              15.00000
 C3              18.00000
 C4              21.00000

AVAILABLE (m/hr):
 WELDING         132.0000
 INST_DOOR       139.5000
 FINISH_PAINTER  178.5000

DEMAND (m/hr)
 C1              45.00000
 C2              65.00000
 C3              90.00000
 C4              65.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
SOLUTION
Global optimal solution found.
Objective value:                                       825.9447
Objective bound:                                       825.9447
Infeasibilities:                                       0.000000

OPTIMAL PRODUCTION PROGRAM:
 PRODUCE:
CONTAINER1 produce:16un x Unit profit: $10.00 = Total: $164.69
CONTAINER2 produce: 2un x Unit profit: $15.00 = Total: $ 33.79
CONTAINER3 produce:15un x Unit profit: $18.00 = Total: $269.48
CONTAINER4 produce:17un x Unit profit: $21.00 = Total: $357.98
 Total:                                                $825.94
```

27

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Factory
- Team

Source:

- Book 3
- Chapter 2.3.19

GOAL

A company has four assembly teams that have been trained to assemble three different product types.

Each team has a specific performance in assembly time (in minutes) of the products.

The product assembled by the teams is evaluated based on their aggregation of value, reliability, elimination of future rework and other elements.

If each assembly team has 1800 minutes of useful work in the week, it is necessary to produce at least 90 units of product 1, 160 of product 2 and 110 of product 3.

Below is all the information necessary for the development of the model.

Performance		Team 1	Team 2	Team 3	Team 4	Demand
Prod 1	min/un	9	5	3	11	90 un
Prod 2	min/un	4	10	7	12	160 un
Prod 3	min/un	9	12	15	16	110 un
Available	min	1,800	1,800	1,800	1,800	-

Value Added		Team 1	Team 2	Team 3	Team 4
Prod 1	$	6.00	4.00	4.00	7.00
Prod 2	$	7.00	10.00	8.00	10.00
Prod 3	$	8.00	10.00	10.00	11.00

Formulate the Linear programming model that optimizes the assembly process.

```
MODEL:
SETS:
PRODUCTS: DEMAND;
RESOURCE: AVAILABLE;
ROUTES( PRODUCTS, RESOURCE): VALUE_ADDED, VOLUME, PERFORMANCE;
TRS( RESOURCE, PRODUCTS): TRA_VOL, TRA_PER, TRA_TOT;
ENDSETS
DATA:
! Products attributes;
PRODUCTS,          DEMAND =
PROD1              90
PROD2              160
PROD3              110;

!  Resources attributes;
RESOURCE,          AVAILABLE =
TEAM1              1800
TEAM2              1800
TEAM3              1800
TEAM4              1800;

! Require          Team1 Team2 Team3 Team4;
VALUE_ADDED   =    6     4     4     7            ! Prod1;
                   7     10    8     10           ! Prod2;
                   8     10    10    11;          ! Prod3;

PERFORMANCE =      9     5     3     11           ! Prod1;
                   4     10    7     12           ! Prod2;
                   9     12    15    16;          ! Prod3;
ENDDATA
SUBMODEL MAX27:
[OBJ] MAX = @SUM( ROUTES( I, J): VALUE_ADDED( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( RESOURCE( J):
        [CAP] @SUM( PRODUCTS( I): PERFORMANCE(I,J) * VOLUME( I, J)) <= AVAILABLE( 1));
! The capacity constraints;
@FOR( PRODUCTS( I):
        [DEM] @SUM( RESOURCE( J): VOLUME( I, J)) <= DEMAND( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Default starting point for variables;
@SET('STARTP',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  VALUE ADDED:", @NEWLINE( 1));
@TABLE(VALUE_ADDED);
@WRITE(" ", @NEWLINE( 1), "  PERFORMANCE min/un:", @NEWLINE( 1));
@TABLE(PERFORMANCE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE min:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND unit:", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX27);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  OPTIMAL PRODUCTION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  ', RESOURCE( J), ' produce:',
        @FORMAT(VOLUME( I, J),'%3.0f'), ' un  ', PRODUCTS( I),' x Unit value: $',
        @FORMAT(VALUE_ADDED (I, J),'%5.2f'), ' = Total: $',
        @FORMAT(VALUE_ADDED (I, J) * VOLUME( I,J),'%7.2f'),
@NEWLINE( 1));
@WRITE('  TOTAL AGGREGATE VALUE:', 36*' ','$', @FORMAT(OBJ,'%7.2f') , @NEWLINE(1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX27);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
VALUE ADDED:
            TEAM1     TEAM2     TEAM3     TEAM4
 PROD1  6.000000  4.000000  4.000000  7.000000
 PROD2  7.000000  10.00000  8.000000  10.00000
 PROD3  8.000000  10.00000  10.00000  11.00000

PERFORMANCE min/un:
            TEAM1     TEAM2     TEAM3     TEAM4
 PROD1  9.000000  5.000000  3.000000  11.00000
 PROD2  4.000000  10.00000  7.000000  12.00000
 PROD3  9.000000  12.00000  15.00000  16.00000

AVAILABLE min:
 TEAM1  1800.000
 TEAM2  1800.000
 TEAM3  1800.000
 TEAM4  1800.000

DEMAND unit:
 PROD1  90.00000
 PROD2  160.0000
 PROD3  110.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                          3380.625
Infeasibilities:                                          0.000000

OPTIMAL PRODUCTION PROGRAM:
TEAM4 produce: 90 un  PROD1 x Unit value: $ 7.00 = Total: $ 630.00
TEAM2 produce:160 un  PROD2 x Unit value: $10.00 = Total: $1600.00
TEAM2 produce: 17 un  PROD3 x Unit value: $10.00 = Total: $ 166.67
TEAM3 produce: 43 un  PROD3 x Unit value: $10.00 = Total: $ 427.08
TEAM4 produce: 51 un  PROD3 x Unit value: $11.00 = Total: $ 556.88
TOTAL AGGREGATE VALUE:                                    $3380.62
```

BLOCK 2

Block: **BLEND**

What foods should people (or animals) use, so that the cost is minimal and they have the nutrients in adequate quantities, and also meet other requirements, such as variety between meals, appearance, taste, etc.?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

GOAL

In this type of problem, the analyst wants to determine levels of use of raw materials in the composition of a food ration.

Restrictions usually relate to desired nutritional characteristics for the finished product, quantities of raw materials and inputs available and demand to be met.

Suppose a problem in which a ration must be elaborated from the mixture of 3 types of grains.

Four nutrients are considered in the final product, as considered in the information below:

Resources / Products			Grain 1	Grain 2	Grain 3	Available
Nutrient	A	kg	2	3	7	1,250
	B	kg	1	1	0	250
	C	kg	5	3	0	900
	D	kg	0.6	0.25	1	233
Cost		$	41.00	35.00	96.00	-

The information that compose the constraints and objective function of the problem presented, serve for the development of the model to minimize costs.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT)  : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        AVAILABLE     =
 NUTR_A           1250
 NUTR_B           250
 NUTR_C           900
 NUTR_D           232.5;
! Products attributes;
 PRODUCT,         COST   =
 GRAIN1           41
 GRAIN2           35
 GRAIN3           96;

! Require p/kg    Grain1      Grain2      Grain3;
 USAGE      =     2           3           7           ! NUTR_A;
                  1           1           0           ! NUTR_B;
                  5           3           0           ! NUTR_C;
                  0.6         0.25        1;          ! NUTR_D;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (kg):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (KG):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST p/kg:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J):'  ',product( J),': ',
      @FORMAT(PRODUCE( J),'%3g'),' Kg  x Unit profit: $',
      @FORMAT(COST( J),'%5.2f'), '  =  Total: $',
      @FORMAT(COST( J) * PRODUCE( J),'%7.2f'),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Blend Model', 'Mix', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (kg):
            GRAIN1     GRAIN2     GRAIN3
 NUTR_A   2.000000   3.000000   7.000000
 NUTR_B   1.000000   1.000000   0.000000
 NUTR_C   5.000000   3.000000   0.000000
 NUTR_D   0.600000   0.250000   1.000000

AVAILABLE (KG):
 NUTR_A  1250.000
 NUTR_B  250.0000
 NUTR_C  900.0000
 NUTR_D  232.5000

COST p/kg:
 GRAIN1  41.00000
 GRAIN2  35.00000
 GRAIN3  96.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              19550.00
Infeasibilities:                              0.000000


IDEAL MIXING PROGRAM:
GRAIN1: 200 Kg  x Unit profit: $41.00  =  Total: $8200.00
GRAIN2:  50 Kg  x Unit profit: $35.00  =  Total: $1750.00
GRAIN3: 100 Kg  x Unit profit: $96.00  =  Total: $9600.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

GOAL

Suppose we want to produce a minimum cost ration by mixing two products A and B, and they have different costs: Product A: $0.30 per kg and Product B: $0.40 per kg.

As for birds, it is known that each requires minimum quantities per week. Described in the formula and will be obtained from these products.

Below are the information necessary for the development of the model.

Resources / Products			Prod A	Prod B	Available
Formula	Vitamin 1	%	5	25	50kg
	Vitamin 2	%	25	10	100kg
	Vitamin 3	%	10	10	60kg
	Vitamin 4	%	35	20	180kg
Cost p/kg		$	0.30	0.40	-

Therefore, to develop a model to obtain the lowest cost in this blend.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,              AVAILABLE   =
 VIT_1                  50
 VIT_2                  100
 VIT_3                  60
 VIT_4                  180;

! Product attributes;
 PRODUCT,               COST        =
 PROD_A                 0.30
 PROD_B                 0.40 ;

! Require (formula) %   PROD_A      PROD_B;
 USAGE   =              0.05        0.25        ! VIT_1;
                        0.25        0.10        ! VIT_2;
                        0.10        0.10        ! VIT_3;
                        0.35        0.20;       ! VIT_4;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST p/kg:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J): '  ',product( J),': ',
      @FORMAT(PRODUCE( J),'%4.1f'),' Kg  x Unit cost: $',
      @FORMAT(COST( J),'%4.2f'), ' =  Total: $',
      @FORMAT(COST( J) * produce( J),'%6.2f'),
@NEWLINE( 1));
! Execute the Graph;
@CHARTPIE( 'Blend Model', 'Mix', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                PROD_A         PROD_B
 VIT_1       0.0500000      0.2500000
 VIT_2       0.2500000      0.1000000
 VIT_3       0.1000000      0.1000000
 VIT_4       0.3500000      0.2000000

AVAILABLE (kg):
 VIT_1       50.00000
 VIT_2       100.0000
 VIT_3       60.00000
 VIT_4       180.0000

COST p/kg:
 PROD_A      0.300000
 PROD_B      0.400000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              190.0000
Infeasibilities:                              0.000000

IDEAL MIXING PROGRAM:
PROD_A: 500.0 Kg  x Unit cost: $0.30 =  Total: $150.00
PROD_B: 100.0 Kg  x Unit cost: $0.40 =  Total: $ 40.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

GOAL

A company that sells agricultural products received an order of 8000 kg of feed. The customer wants it to have at least 20% corn, 15% grain and 15% mineral salts.

Resources / Products			R1	R2	R3	R4	Available (kg)
Formula	Corn	%	30	5	20	10	1,660
	Grains	%	10	30	15	10	1,200
	Minerals	%	20	20	20	30	1,200
Cost p/kg			250.00	300.00	320.00	150.00	-

What is the ideal blend in order to minimize the cost?

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
  RESOURCE,          AVAILABLE  =
  CORN               1600
  GRAINS             1200
  MINERALS           1200;
! Products attributes;
  PRODUCT,           COST   =
  R1                 250
  R2                 300
  R3                 320
  R4                 150;

! Require (Formula) %   R1      R2      R3      R4;
  USAGE           =     0.30    0.05    0.20    0.10    ! CORN;
                        0.10    0.30    0.15    0.10    ! GRAINS;
                        0.20    0.20    0.20    0.30;   ! MINERALS;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( PRODUCT( p):COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST p/kg:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE(J) #GT# 0: '  ',product( J),': ',
       @FORMAT(PRODUCE( J),'%4.1f'),'Kg x Unit cost: $',
       @FORMAT(COST( J),'%4.2f'), ' = Total: $',
       @FORMAT(COST( J) * produce( J),'%10.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                      R1          R2             R3             R4
 CORN           0.3000000   0.0500000      0.2000000      0.1000000
 GRAINS         0.1000000   0.3000000      0.1500000      0.1000000
 MINERALS       0.2000000   0.2000000      0.2000000      0.3000000

AVAILABLE (kg):
 CORN           1600.000
 GRAINS         1200.000
 MINERALS       1200.000

COST p/kg:
 R1             250.0000
 R2             300.0000
 R3             320.0000
 R4             150.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                   1941176.
Infeasibilities:                                   0.000000

IDEAL MIXING PROGRAM:
R1: 4941.2Kg x Unit cost: $250.00 = Total: $1235294.12
R2: 2352.9Kg x Unit cost: $300.00 = Total: $ 705882.35
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

GOAL

The weekly dietary needs of a particular animal correspond to the formula described below, as well as the availability of proteins and carbohydrates and costs of each type of feed:

RESOURCES / PRODUCTS			RA	RB	RC	RD	RE	AVAILABLE
FORMULA	PROTEINS	%	25	25	45	35	25	200 kg
	CARBOHYDRATES	%	55	20	10	35	20	250 kg
COST p/kg		$	3.00	2.00	4.00	3.00	3.00	-

What mixture of these rations satisfies dietary requirements at the lowest cost to the owner?

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
  RESOURCE,                AVAILABLE    =
  PROTEINS                 200
  CARBOHYDRATES            250;
! Products attributes;
  PRODUCT,                 COST   =
  RA                       3.00
  RB                       2.00
  RC                       4.00
  RD                       3.00
  RE                       3.00;

! Require ( formula) %     RA     RB     RC     RD     RE;
  USAGE       =            0.25    0.25  0.45   0.35   0.25   !  PROTEINS;
                           0.55    0.20  0.10   0.35   0.20;  !  CARBOHYDRATES;
ENDDATA
SUBMODEL MIN4:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST per kg:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN4);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE( J) #GT# 0: '    Product: ',product( J),'  ',
        @FORMAT(PRODUCE( J),'%6.1f'),' kg  x cost p/kg: $',
        @FORMAT(COST( J),'%4.2f'), ' =  total: $',
        @FORMAT(COST( J) * produce( J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                      RA        RB        RC        RD        RE
 PROTEINS       0.250000  0.250000  0.450000  0.350000  0.250000
 CARBOHYDRATES  0.550000  0.200000  0.100000  0.350000  0.200000

AVAILABLE (kg):
 PROTEINS       200.000
 CARBOHYDRATES  250.000

COST per kg:
 RA  3.00000
 RB  2.00000
 RC  4.00000
 RD  3.00000
 RE  3.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                     1857.14
Infeasibilities:                                     0.00000

IDEAL MIXING PROGRAM:
  Product: RA   166.7 kg  x cost p/kg: $3.00 =  total: $ 500.00
  Product: RD   452.4 kg  x cost p/kg: $3.00 =  total: $1357.14
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

Source:

- Book 5
- Page 259

GOAL

A farmer is raising pigs for market and wishes to determine the quantity of the available types of feed that should be given to each pig to meet certain nutritional requirements at minimum cost.

The units of each type of basic nutritional ingredient contained in a pound of each feed type is given in the following table along with the daily nutritional requirement and feed costs.

Resources / Products		Corn	Tankage	Alfalfa	Required
Carbohydrates	pound	9	2	4	20 un
Proteins	pound	3	8	6	18 un
Vitamins	pound	1	2	6	15 un
Cost (cents)/pound	$	7	6	5	-

```
MODEL:
SETS:
 PRODUCT : COST;
 RESOURCE: REQUIRED;
 RXP( RESOURCE, PRODUCT) : FORMULA, PRODUCE;
ENDSETS
DATA:
! Resource attributes;
  RESOURCE,              REQUIRED    =
  CARBOHYDRATES          20
  PROTEINS               18
  VITAMINS               15;

! Products attributes;
  PRODUCT,               COST   =
  CORN                   7.00
  TANKAGE                6.00
  ALFALFA                5.00;

! Require                CORN        TANKAGE      ALFALFA;
  FORMULA      =         9           2            4            !  CARBOHYDRATES;
                         3           8            6            !  PROTEINS;
                         1           2            6;           !  VITAMINS;
ENDDATA
SUBMODEL MIN5:
[OBJ] MIN = @SUM( RXP( I,J): (COST( J) * FORMULA(I,J)) * PRODUCE( I,J));
! The Available constraints;
@FOR( RESOURCE( I):
        [AVA] @SUM( PRODUCT( J): PRODUCE(I, J )) >= REQUIRED( I););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (pound):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  REQUIRED (pound):", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  COST (cents)/lb.", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN5);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I,J)| PRODUCE( I, J) #GT# 0: '  ',
        @FORMAT(RESOURCE(I),'-14s'),' ',
        @FORMAT(PRODUCT( J),'-7s'),' ',
        @FORMAT(PRODUCE( I, J),'%2.0f'),' pound x cost: $',
        @FORMAT(COST( J)* FORMULA(I,J),'%5.2f'), ' = total: $',
        @FORMAT(COST( J)* FORMULA(I,J) * PRODUCE( I,J),'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (pound):
                    CORN     TANKAGE    ALFALFA
 CARBOHYDRATES  9.000000  2.000000  4.000000
 PROTEINS       3.000000  8.000000  6.000000
 VITAMINS       1.000000  2.000000  6.000000

REQUIRED (pound):
 CARBOHYDRATES  20.00000
 PROTEINS       18.00000
 VITAMINS       15.00000

COST (cents)/lb.
 CORN     7.000000
 TANKAGE  6.000000
 ALFALFA  5.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                         723.0000
Infeasibilities:                                         0.000000


IDEAL MIXING PROGRAM:
CARBOHYDRATES  TANKAGE 20 pound x cost: $12.00 = total: $240.00
PROTEINS       CORN    18 pound x cost: $21.00 = total: $378.00
VITAMINS       CORN    15 pound x cost: $ 7.00 = total: $105.00
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula

Source:

- Book 1
- Page 50

GOAL

An ice-cream maker wants to produce 100 kg of ice cream at a minimum cost, whose composition in the details, in addition to the raw materials available.

Components / Formula	Cost	FAT	SLNG	TSL	SUGAR	TS	WATER	STAB	EMUL
	$	%	%	%	%	%	%	%	%
Cream 40%	27.00	40	5.4	45.4	-	45.4	54.6	-	-
Cream 38%	26.00	38	5.6	43.6	-	43.6	56.4	-	-
Milk 3,2%	3.00	3.2	8.7	11.9	-	11.9	88.1	-	-
Milk 4,0%	3.00	4.0	8.6	12.6	-	12.6	87.4	-	-
Fatty Condensed Milk	7.00	8	20	28	-	28	72	-	-
Lean Condensed Skin	3.00	-	28	28	-	28	72	-	-
Butter	15.00	5	92	97	-	97	3	-	-
Whey Dried Solids	10.00	-	95	95	-	95	5	-	-
Saccharose	10.00	-	-	-	100	100	-	-	-
Cane Broth	9.00	-	-	-	67	67	33	-	-
Stabilizer	55.00	-	-	-	-	80	20	-	-
Emulsifier	78.00	-	-	-	-	-	-	-	-
Water	0.00	-	-	-	-	-	100	-	-
Requirements	MIN	10	10.5	20.5	11	32.5	58.5	0.37	0.1
	MAX	16	13	25	17	41.5	62.5		

```
MODEL:
SETS:
 PRODUCTS: COST, PRODUCE;
 RESOURCE: REQ_MIN, REQ_MAX;
 XYZ( PRODUCTS, RESOURCE): U1;
 RXP(RESOURCE, PRODUCTS) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        REQ_MIN,          REQ_MAX  =
 FAT              10                16
 SLNG             10.5              13
 TSL              20.5              25
 SUGAR            11                17
 TS               32.5              41.5
 WATER            58.5              62.5;

! Products attributes;
 PRODUCTS,        COST  =
 CREAM40          27    ! Cream 40%  ;
 CREAM38          26    ! Cream 38%;
 MILK32           3     ! Milk 3.2%;
 MILK40           3     ! Milk 4.0%;
 FATCONDMILK      7     ! Fatty condensed milk;
 LEACONDMILK      3     ! Lean condensed milk;
 BUTTER           15    ! Butter;
 WHEY             10    ! Whey proteins;
 SACCHAROSE       10
 CANE_BROTH       9
 STABILIZER       55
 EMULSIFIER       78
 WATER            0;

! Require (formula) %;
 USAGE   =
 0.4         0.38  0.032 0.04  0.08  0     0.5   0     0     0     0     0     0    !FAT;
 0.054       0.056 0.087 0.086 0.20  0.28  0.92  0.95  0     0     0     0     0    !SLNG;
 0.454       0.436 0.119 0.126 0.28  0.28  0.97  0.95  0     0     0     0     0    !TSL;
 0           0     0     0     0     0     0     0     1.0   0.67  0     0     0    ! SUGAR;
 0.454       0.436 0.119 0.126 0.28  0.28  0.97  0.95  1.0   0.67  0.8   0     0    !TS;
 0.546       0.564 0.881 0.874 0.72  0.72  0.03  0.05  0     0.33  0.20  0     1.0; !WATER;
ENDDATA
SUBMODEL MIN6:
[OBJ]MIN = @SUM( PRODUCTS( p): COST( p) * PRODUCE( p));
! 100kg of ice cream;
@SUM( PRODUCTS( p): PRODUCE(p)) = 100;
! The minimum required constraints;
@FOR( RESOURCE( r):
        [RMIN] @SUM( PRODUCTS( c): USAGE( r, c) * PRODUCE( c )) >= REQ_MIN( r));
! The maximum required constraints;
@FOR( RESOURCE( row):
        [RMAX] @SUM( PRODUCTS( col): USAGE( row, col) * PRODUCE( col )) <= REQ_MAX( row););
! Min/Max Sugar;
USAGE(4,9)  * PRODUCE(9)   * USAGE(4,10)  * PRODUCE(9)  >= 11;
USAGE(4,9)  * PRODUCE(10)  * USAGE(4,10)  * PRODUCE(10) <= 17;
! min and max Stabilizer and Emulsifier;
PRODUCE(11)  = 0.37;
PRODUCE(12)  = 0.1;
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
!Precision in digits for standard solution reports;
@SET('PRECIS',5);
! Set page width;
@SET('LINLEN',120);
!Data block;
U1 = @TRANSPOSE(USAGE);
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(U1);
@WRITE(" ", @NEWLINE( 1), "  REQUIREMENTS_MIN (%):", @NEWLINE( 1));
@TABLE(REQ_MIN);
@WRITE(" ", @NEWLINE( 1), "  REQUIREMENTS_MAX (%):", @NEWLINE( 1));
@TABLE(REQ_MAX);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN6);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCTS( J) |  produce(j)  #GT# 0: '  Required: ',
        @FORMAT(PRODUCTS( J),'-11s'),
        @FORMAT(PRODUCE( J),'%8.4f'),'% x Cost: $',
        @FORMAT(COST( J),'%5.2f'), ' = TOTAL: $',
        @FORMAT(COST( J) * PRODUCE( J),'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

DATA:
FORMULA (%):

	FAT	SLNG	TSL	SUGAR	TS	WATER
CREAM40	0.40000	0.05400	0.45400	0.0000	0.45400	0.54600
CREAM38	0.38000	0.05600	0.43600	0.0000	0.43600	0.56400
MILK32	0.03200	0.08700	0.11900	0.0000	0.11900	0.88100
MILK40	0.04000	0.08600	0.12600	0.0000	0.12600	0.87400
FATCONDMILK	0.08000	0.20000	0.28000	0.0000	0.28000	0.72000
LEACONDMILK	0.0000	0.28000	0.28000	0.0000	0.28000	0.72000
BUTTER	0.50000	0.92000	0.97000	0.0000	0.97000	0.03000
WHEY	0.0000	0.95000	0.95000	0.0000	0.95000	0.05000
SACCHAROSE	0.0000	0.0000	0.0000	1.0000	1.0000	0.0000
CANE_BROTH	0.0000	0.0000	0.0000	0.67000	0.67000	0.33000
STABILIZER	0.0000	0.0000	0.0000	0.0000	0.80000	0.20000
EMULSIFIER	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
WATER	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

COST:

CREAM40	27.000
CREAM38	26.000
MILK32	3.0000
MILK40	3.0000
FATCONDMILK	7.0000
LEACONDMILK	3.0000
BUTTER	15.000
WHEY	10.000
SACCHAROSE	10.000
CANE_BROTH	9.0000
STABILIZER	55.000
EMULSIFIER	78.000
WATER	0.0000

REQUIREMENTS_MIN (%):

FAT	10.000
SLNG	10.500
TSL	20.500
SUGAR	11.000
TS	32.500
WATER	58.500

REQUIREMENTS_MAX (%):

FAT	16.000
SLNG	13.000
TSL	25.000
SUGAR	17.000
TS	41.500
WATER	62.500

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Local optimal solution found.
Objective value:                                    802.79
Infeasibilities:                                    0.0000

IDEAL MIXING PROGRAM:
Required: CREAM40      11.6719% x Cost: $27.00 = TOTAL: $315.14
Required: MILK40       63.8368% x Cost: $ 3.00 = TOTAL: $191.51
Required: BUTTER        5.5556% x Cost: $15.00 = TOTAL: $ 83.33
Required: WHEY          1.8617% x Cost: $10.00 = TOTAL: $ 18.62
Required: SACCHAROSE   16.6040% x Cost: $10.00 = TOTAL: $166.04
Required: STABILIZER    0.3700% x Cost: $55.00 = TOTAL: $ 20.35
Required: EMULSIFIER    0.1000% x Cost: $78.00 = TOTAL: $  7.80

SLACK/SURPLUS LIMIT = MINIMUM REQUIREMENTS:
             PROD     LIMIT     VALUE
 FAT       10.000    10.000    0.0000
 SLNG      8.0000    10.500   -2.5000
 TSL       20.500    20.500    0.0000
 SUGAR     5.3960    11.000   -5.6040
 TS        27.600    32.500   -4.9000
 WATER     54.500    58.500   -4.0000
```

7

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

Source:

- Book 9
- Page 18

GOAL

In a farm you want to make **10000 kilos of feed** at the lowest possible cost.

According to the recommendations of the veterinarian of the farm animals the same should contain: 15% protein, minimum 8% fiber, minimum 1100 calories per kilo and maximum 2250 calories per kilo.

To make the ration, 4 ingredients are available whose technical-economic characteristics are shown in the details (data in%, except calories and costs).

The ration must be made containing at least 20% of maize and at most 12% of soya.

Nutritional Requirement			Barley	Oats	Soya	Corn		Min	Max
Formula	Proteins	%	6.9	8.5	9.0	27.1	kg	1500	
	Fibers	%	6	11	11	14	kg	800	
	Calories /kg	cal	1,760	1,700	1,056	1,400	cal	1,100,000	2,250,000
	Limit	kg			1,200	2,000	-		
Cost p/kg		$	43.00	48.00	44.00	56.00	-		

Therefore, formulate a PL model for the problem.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE_MIN, AVAILABLE_MAX;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Product attributes;
 PRODUCT ,          COST   =
 BARLEY             30
 OATS               48
 SOYA               44
 CORN               56;
! Resources attributes;
 RESOURCE,          AVAILABLE_MIN,      AVAILABLE_MAX  =
 PROTEINS           1500                1500
 FIBERS             800                 1500
 CALORIES           11000000            22500000;
! Required ( formula ) Barley      Oats        Soya        Corn   ;
 USAGE  =           0.069       0.085       0.090       0.271   ! PROTEINS (%);
                    0.060       0.110       0.110       0.140   ! FIBERS (%);
                    1760        1700        1056        1400;   ! CALORIES (CAL);
ENDDATA
SUBMODEL MIN7:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! Amount of feed;
@SUM( PRODUCT( p): PRODUCE( p)) = 10000;
PRODUCE( 3) <= 1200;                                ! Soya ( 12%);
PRODUCE( 4) >= 2000;                                ! Corn ( 20%);
! The minimum available constraints;
@FOR( RESOURCE( r):
        [AVA_MIN] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE_MIN( r););
! The maximum available  constraints;
@FOR( RESOURCE( r):
         [AVA_MAX] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE_MAX(r));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%,CAL):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE_MIN (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE_MIN);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE_MAX (kg):", @NEWLINE( 1));
@TABLE(AVAILABLE_MAX);
@WRITE(" ", @NEWLINE( 1), "  COST p/kg:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN7);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J),'-7s'),' ',
        @FORMAT(PRODUCE( J),'%6.1f'),'Kg x Unit cost: $',
        @FORMAT(COST( J), '%5.2f'),' = Total: $',
        @FORMAT(COST( J) * PRODUCE( J),'%9.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%,CAL):
                  BARLEY          OATS          SOYA          CORN
 PROTEINS      0.0690000    0.08500000    0.09000000    0.27100000
 FIBERS        0.0600000    0.11000000    0.11000000    0.14000000
 CALORIES      1760.0000     1700.0000     1056.0000     1400.0000

AVAILABLE_MIN (kg):
 PROTEINS    1500.0000
 FIBERS      800.00000
 CALORIES    11000000.

AVAILABLE_MAX (kg):
 PROTEINS  1500.0000
 FIBERS    1500.0000
 CALORIES  22500000.

COST p/kg:
 BARLEY   30.000000
 OATS     48.000000
 SOYA     44.000000
 CORN     56.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    404257.43
Infeasibilities:                                    0.0000000
IDEAL MIXING PROGRAM:
BARLEY   5990.1 Kg x Unit cost: $30.00 = Total: $179702.97
CORN     4009.9 Kg x Unit cost: $56.00 = Total: $224554.46
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Ration
- Formula

Source:

- Book 2
- Page 222

GOAL

A dog food company produces two types of feed: A and B. Cereals and meat are used for the manufacture of feed, according to the table below:

Resources/Products		A	B	Available	Cost
Cereals	lb.	5	2	30,000	1.00
Meat	lb.	1	4	10,000	4.00
Price (Pac, 6 lb.)	lb.	20.00	30.00		-

How much of each feed is needed to maximize profit?

```
MODEL:
SETS:
 RESOURCES: AVAILABLE, COST;
 PRODUCTS: PRICE, PRODUCE;
 RXP( RESOURCES, PRODUCTS): FORMULA;
ENDSETS
DATA:
! Products attributes;
 PRODUCTS        PRICE =
 A               20
 B               30;
! Brands attributes;
 RESOURCES,      AVAILABLE,   COST    =
 CEREALS         30000        1
 MEAT            10000        4;
! Required       A            B;
 FORMULA   =     5            2       ! CEREALS;
                 1            4;      ! MEAT;
ENDDATA
SUBMODEL MAX8:
[OBJ] MAX = ((PRICE(1) - COST_C ) * PRODUCE(1)) + ((PRICE(2) - COST_M) * PRODUCE(2));
!Cost of cereals;
COST_C = FORMULA(1,1) * COST(1) + FORMULA(2,1) * COST(2);
!Cost of Meat;
COST_M = FORMULA(1,2) * COST(1) + FORMULA(2,2) * COST(2);
! The Available Constraints;
@FOR(RESOURCES( I):
      @SUM(PRODUCTS(J): FORMULA(I,J) * PRODUCE(J)) <= AVAILABLE(I););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA - Resources vs Products (lb.):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (lb.):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE (p/lb.):", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  COST (p/lb):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX8);
! Solution Report;
@WRITE( "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCTS(K): '  Product:',
      @FORMAT(PRODUCTS( K),'4s'),', ',
      @FORMAT(PRODUCE( K),'%8.3f'),' lb. x Cost: $',
      @FORMAT(@IF(K #EQ# 1, COST_C, COST_M),'%5.2f'), '  = Total:   $',
      @FORMAT(@IF(K # EQ# 1, COST_C * PRODUCE(K), COST_M * PRODUCE(K)),'%9.2f'), @NEWLINE(1),44*' ',' - Revenue: $',
      @FORMAT(@IF(K  #EQ# 1, PRICE(1) * PRODUCE(K), PRICE(2) * PRODUCE(K)),'9.2f'), @NEWLINE(1),44*' ',' = Profit:  $',
      @FORMAT(@IF(K  #EQ# 1, PRICE(1) * PRODUCE(K) - COST_C * PRODUCE(K),
                      PRICE(2) * PRODUCE(K) - COST_M * PRODUCE(K)), '%9.2f'),
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX8);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA - resources vs Products (lb.):
                 A          B
 CEREALS  5.000000  2.000000
 MEAT     1.000000  4.000000

AVAILABLE (lb.):
 CEREALS  30000.00
 MEAT     10000.00

PRICE (p/lb.):
 A  20.00000
 B  30.00000

COST (p/lb):
 CEREALS  1.000000
 MEAT     4.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    74444.44
Infeasibilities:                                    0.000000

IDEAL MIXING PROGRAM:
Product:   A, 5555.556 lb. x Cost: $ 9.00  = Total:   $ 50000.00
                                           - Revenue: $111111.11
                                           = Profit:  $ 61111.11

Product:   B, 1111.111 lb. x Cost: $18.00  = Total:   $ 20000.00
                                           - Revenue: $ 33333.33
                                           = Profit:  $ 13333.33
```

9

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula

Source:

- Book 4
- Chess

GOAL

In blending problems, two or more raw materials are to be blended into one or more finished goods, satisfying one or more quality requirements on the finished goods. In this example, we blend mixed nuts into four different brands with a goal of maximizing revenue.

The Chess Snackfoods Co. markets four brands of mixed nuts. The four brands of nuts are called Pawn, Knight, Bishop, and King. Each brand contains a specified ratio of peanuts and cashews.

The table below lists the number of ounces of the two nuts contained in each pound of each brand and the profit the company receives per pound of each brand.

Resources/Products			Pawn	Knight	Bishop	King	Available
Formula	Peanut	oz	15	10	6	2	750 pounds
	Cashew Nut	oz	1	6	10	14	250 pounds
Price		$	2.00	3.00	4.00	5.00	-

Chess has contracts with suppliers to receive 750 pounds of peanuts/day and 250 pounds of cashews/day.

Our problem is to determine the number of pounds of each brand to produce each day to maximize total revenue without exceeding the available supply of nuts.

```
MODEL:
SETS:
  NUTS: SUPPLY;
  BRANDS: PROFIT, PRODUCE;
  RXP( NUTS, BRANDS): FORMULA;
ENDSETS
DATA:
! Nuts attributes;
  NUTS,            SUPPLY      =
  PEANUTS          750
  CASHEWS          250;

! Brands attributes;
  BRANDS,          PROFIT    =
  PAWN             2
  KNIGHT           3
  BISHOP           4
  KING             5;

! Required         Pawn        Knight      bishop      King;
  FORMULA    =     15          10          6           2           ! Peanuts;
                   1           6           10          14;         ! Cashews;
ENDDATA
SUBMODEL MAX9:
[OBJ] MAX = @SUM( BRANDS( I):PROFIT( I) * PRODUCE( I));
! The Supply Constraints;
@FOR( NUTS( I):
        [SUP] @SUM( BRANDS( J):FORMULA( I, J) * PRODUCE( J) / 16) <= SUPPLY( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (oz):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (pounds):", @NEWLINE( 1));
@TABLE(SUPPLY);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (p/oz):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX9);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( BRANDS( J)| PRODUCE( J) #GT# 0: '  Brand:',
        @FORMAT(BRANDS( J),'4s'),', ',
        @FORMAT(PRODUCE( J),'%6.2f'),' oz x Unit profit: $',
        @FORMAT(PROFIT( J),'%4.2f'), ' = Total: $',
        @FORMAT(PROFIT( J) * PRODUCE( J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN9);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (oz):
                PAWN         KNIGHT        BISHOP         KING
 PEANUTS  15.00000       10.00000      6.000000      2.000000
 CASHEWS  1.000000       6.000000      10.00000      14.00000

AVAILABLE (pounds):
 PEANUTS  750.0000
 CASHEWS  250.0000

PROFIT (p/oz):
 PAWN    2.000000
 KNIGHT  3.000000
 BISHOP  4.000000
 KING    5.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    2692.308
Infeasibilities:                                    0.000000

IDEAL MIXING PROGRAM:
Brand:PAWN, 769.23 oz x Unit profit: $2.00 = Total: $1538.46
Brand:KING, 230.77 oz x Unit profit: $5.00 = Total: $1153.85
```

BLOCK 3

Block: FINANCE

What actions should be included in a portfolio of investments so that the profit is maximum and the forecasts of profitability and government restrictions are respected?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Business
- Resells
- Purchase

GOAL

A company buys and resells a product that can be purchased at the following cost:

Resources / Periods			M1	M2	M3	M4	M5	M6	Capacity
Cost	Supplier A	$	1.20	1.30	1.40	1.50	1.60	1.70	150 un
	Supplier B	$	1.30	1.40	1.50	1.60	1.70	1.80	90 un
	Stockholm	$	0.05	0.05	0.05	0.05	0.05	0.05	50 un
Demand		un	120	150	120	200	150	180	

The company wants to minimize the total cost of operations for 6 months, which is the sum of the cost of purchase and the cost of storage.

The sales forecast for the next 6 months is: 120,150,120,200,150 and 180 unity.

The storage cost from one period to another is $0.05 and the stock at the beginning of month 1 is zero.

What is the monthly scheme of purchases that implies total cost.

Consider the cost of any given month as the sum of the costs of purchasing the product from the two suppliers plus the cost of the existing inventory at the end of the period.

```
MODEL:
SETS:
 PERIOD  : DEMAND, COST_TT;
 RESOURCE: CAPACITY ;
 RXP( PERIOD, RESOURCE) : COST, PRODUCE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE ,      CAPACITY      =
 SUPPLIER_A      150
 SUPPLIER_B      90
 STOCK           50;
! Period attributes;
 PERIOD,         DEMAND        =
 MONTH1          120
 MONTH2          150
 MONTH3          120
 MONTH4          200
 MONTH5          150
 MONTH6          180;
! Costs          Supplier_A   Supplier_B   Stock;
 COST      =     1.20         1.30         0.05          ! MONTH1;
                 1.30         1.40         0.05          ! MONTH2;
                 1.40         1.50         0.05          ! MONTH3;
                 1.50         1.60         0.05          ! MONTH4;
                 1.60         1.70         0.05          ! MONTH5;
                 1.70         1.80         0.05;         ! MONTH6;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM(PERIOD(J): COST_TT(J));
! Cost calculation;
@FOR(PERIOD(I):COST_TT(I) = @SUM(RESOURCE(J): COST(I,J) * PRODUCE(I,J)););
! The demand constraint;
@FOR( RXP(I,J) | J #EQ# 1: PRODUCE(I,1) + PRODUCE(I,2) - PRODUCE(I,3) >= DEMAND(I));
! The capacity constraint;
@FOR( PERIOD(I): @FOR( RESOURCE(J): PRODUCE(I,J) <= CAPACITY(J)););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE(" DATA:", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST, 1);
@WRITE(" ", @NEWLINE( 1), "  CAPACITY (un):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL OPERATION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) | PRODUCE(I,J) #GT# 0: '   . ',
        @FORMAT(PERIOD(I),'-6s'),', ',
        @FORMAT(RESOURCE(J),'-10s'), '  Provide:',
        @FORMAT(PRODUCE(I,J),'%4.0f'),' un x cost: $',
        @FORMAT(COST( I,J),'%5.2f'), ' = Total: $',
        @FORMAT(COST( I,J) * PRODUCE( I,J),'%7.2f'),
@NEWLINE( 1));
@WRITE('   TOTAL COST:',51*' ','$', @FORMAT(OBJ,'%7.2f'),@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 COST:
            SUPPLIER_A   SUPPLIER_B       STOCK
  MONTH1        1.2000       1.3000    0.050000
  MONTH2        1.3000       1.4000    0.050000
  MONTH3        1.4000       1.5000    0.050000
  MONTH4        1.5000       1.6000    0.050000
  MONTH5        1.6000       1.7000    0.050000
  MONTH6        1.7000       1.8000    0.050000

 CAPACITY (un):
  SUPPLIER_A    150.00
  SUPPLIER_B    90.000
  STOCK         50.000

 DEMAND (un):
  MONTH1        120.00
  MONTH2        150.00
  MONTH3        120.00
  MONTH4        200.00
  MONTH5        150.00
  MONTH6        180.00
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                  1361.0
Infeasibilities:                                                  0.0000

IDEAL OPERATION PROGRAM:
 . MONTH1, SUPPLIER_A  Provide: 120 un x cost: $ 1.20 = Total: $ 144.00
 . MONTH2, SUPPLIER_A  Provide: 150 un x cost: $ 1.30 = Total: $ 195.00
 . MONTH3, SUPPLIER_A  Provide: 120 un x cost: $ 1.40 = Total: $ 168.00
 . MONTH4, SUPPLIER_A  Provide: 150 un x cost: $ 1.50 = Total: $ 225.00
 . MONTH4, SUPPLIER_B  Provide:  50 un x cost: $ 1.60 = Total: $  80.00
 . MONTH5, SUPPLIER_A  Provide: 150 un x cost: $ 1.60 = Total: $ 240.00
 . MONTH6, SUPPLIER_A  Provide: 150 un x cost: $ 1.70 = Total: $ 255.00
 . MONTH6, SUPPLIER_B  Provide:  30 un x cost: $ 1.80 = Total: $  54.00
 TOTAL COST:                                                     $1361.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Business
- NPV

Source:

- Book 2
- Page 182

GOAL

The technical head of a research institute has received 18 new project proposals from its engineers and analysts.

Five of them were selected as being of interest because they agreed with the institute's research lines. However, the institute does not have enough cash to carry out these projects simultaneously.

Therefore, it is necessary to indicate the projects that must be executed in order to maximize the NPV. The cash flows for each of the selected projects are shown below :

Cash Flows			Capital Necessary				
Projects	VPL		2005	2006	2007	2008	2009
P1	$	141	75	25	20	15	10
P2	$	187	90	35	0	0	30
P3	$	121	60	15	15	15	15
P4	$	83	30	20	10	5	5
P5	$	265	100	25	20	20	20
P6	$	127	50	20	10	30	40
Available	$	-	250	75	50	50	50

```
MODEL:
SETS:
 PERIOD : AVAILABLE;
 PROJECT:  PRODUCE, NPV;
 RXP( PROJECT, PERIOD) : USAGE;
ENDSETS
DATA:
! Periods attributes;
 PERIOD,              AVAILABLE          =
 Y1                   250
 Y2                   75
 Y3                   50
 Y4                   50
 Y5                   50;

! Projects attributes;
 PROJECT,             NPV      =
 P1                   141
 P2                   187
 P3                   121
 P4                   83
 P5                   265
 P6                   127;

! Required            Y1     Y2     Y3     Y4     Y5;
 USAGE    =           75     25     20     15     10      ! P1;
                      90     35     0      0      30      ! P2;
                      60     15     15     15     15      ! P3;
                      30     20     10     5      5       ! P4;
                      100    25     20     20     20      ! P5;
                      50     20     10     30     40;     ! P6;
ENDDATA
SUBMODEL MAX2:
[OBJ] MAX = @SUM( PROJECT( p): NPV( p) * PRODUCE( p));
! The available constraint;
@FOR( PERIOD( L):
      [AVA] @SUM( PROJECT( C): USAGE( C,L) * PRODUCE( C)) <= AVAILABLE( L););
@FOR(PROJECT(P): @BIN(PRODUCE(P)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  CAPITAL NECESSARY:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  CAPITAL AVAILABLE:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  NPV:", @NEWLINE( 1));
@TABLE(NPV);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL OPERATION PROGRAM: ", @NEWLINE( 1));
!@WRITE('  SELECTED PROJECTS:', @NEWLINE(1));
@WRITEFOR( PROJECT( J)|PRODUCE(J) #EQ# 1: '  . ',
      @FORMAT(PROJECT( J),'-3s'), ' Year: ', PERIOD(J), '  =  Net Present Value:',' $',
      @FORMAT(NPV( J) * PRODUCE(J),'%6.2f'),
@NEWLINE( 1));
@WRITE('  TOTAL:',32*' ','$', @FORMAT(OBJ,'%6.2f'), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
CAPITAL NECESSARY:
           Y1       Y2       Y3       Y4       Y5
 P1  75.0000  25.0000  20.0000  15.0000  10.0000
 P2  90.0000  35.0000  0.00000  0.00000  30.0000
 P3  60.0000  15.0000  15.0000  15.0000  15.0000
 P4  30.0000  20.0000  10.0000  5.00000  5.00000
 P5  100.000  25.0000  20.0000  20.0000  20.0000
 P6  50.0000  20.0000  10.0000  30.0000  40.0000

CAPITAL AVAILABLE:
 Y1  250.000
 Y2  75.0000
 Y3  50.0000
 Y4  50.0000
 Y5  50.0000

NPV:
 P1  141.000
 P2  187.000
 P3  121.000
 P4  83.0000
 P5  265.000
 P6  127.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                       489.000
Objective bound:                       489.000
Infeasibilities:                       0.00000
Total solver iterations:                    70

IDEAL OPERATION PROGRAM:
SELECTED PROJECTS:
. P1  Year: Y1  =  Net Present Value: $141.00
. P4  Year: Y4  =  Net Present Value: $ 83.00
. P5  Year: Y5  =  Net Present Value: $265.00
TOTAL:                                 $489.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Business

Source:

- Book 4
- Costing

GOAL

In this example the expectation is to obtain the lowest cost project, considering the different attributes of each one, such as: number of beds, bath, footage and cost.

The information used in the model is described below:

Projects	Beds	Baths	SQFT	Cost
P1	5	4	6,200	559,608
P2	2	1	820	151,826
P3	1	1	710	125,943
P4	4	3	4,300	420,801
P5	4	2	3,800	374,751
P6	3	1	2,200	251,674
P7	3	2	3,400	332,426

The model needs to estimate a value for each project that will be used in the objective (est) function.

```
MODEL:
SETS:
  PROJECTS: BEDS , BATHS, SQFT, COST, EST;
ENDSETS
DATA:
PROJECTS,       BEDS,       BATHS,      SQFT,       COST  =
P1              5           4           6200        559608
P2              2           1           820         151826
P3              1           1           710         125943
P4              4           3           4300        420801
P5              4           2           3800        374751
P6              3           1           2200        251674
77              3           2           3400        332426;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @MAX( PROJECTS: @ABS( COST - EST));
@FOR( PROJECTS:  EST = A0 + A1 * BEDS + A2 * BATHS + A3 * SQFT);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "   BATHS:", @NEWLINE( 1));
@TABLE(BATHS);
@WRITE(" ", @NEWLINE( 1), "   BEDS:", @NEWLINE( 1));
@TABLE(BEDS);
@WRITE(" ", @NEWLINE( 1), "   SQUARE FEET:", @NEWLINE( 1));
@TABLE(SQFT);
@WRITE(" ", @NEWLINE( 1), "   PROJECT COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1));
@WRITEFOR( PROJECTS( J) : '  ',
        @FORMAT(PROJECTS(J),'-3s'),' Baths:',
        @FORMAT(BATHS( J),'%1.0f'),' Beds:',
        @FORMAT(BEDS( J), '%1.0f'),' Square feet:',
        @FORMAT(SQFT( J), '%4.0f'),' Cost: $',
        @FORMAT(COST( J), '%6.0f'),' - Est: $',
        @FORMAT(EST( J),  '%6.0f'),' = Objective: $',
        @FORMAT(COST( J) - EST( J), '%9.3f'),
        @IF( @MIN(PROJECTS(J): SQFT(J)) #EQ# SQFT(J), ' Selected', ' No '),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE(1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 BATHS:
 P1  4.000000
 P2  1.000000
 P3  1.000000
 P4  3.000000
 P5  2.000000
 P6  1.000000
 P7  2.000000

 BEDS:
 P1  5.000000
 P2  2.000000
 P3  1.000000
 P4  4.000000
 P5  4.000000
 P6  3.000000
 P7  3.000000

 SQUARE FEET:
 P1  6200.000
 P2  820.0000
 P3  710.0000
 P4  4300.000
 P5  3800.000
 P6  2200.000
 P7  3400.000

 PROJECT COST:
 P1  559608.0
 P2  151826.0
 P3  125943.0
 P4  420801.0
 P5  374751.0
 P6  251674.0
 P7  332426.0
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                    1426.660
Objective bound:                                                    1426.660
Infeasibilities:                                                    0.000000
Total solver iterations:                                                  34

OPTIMAL PLANNING PROGRAM:
P1  Baths:4 Beds:5 Square feet:6200 Cost: $559608 - Est: $561035 = Objective: $-1426.660 No
P2  Baths:1 Beds:2 Square feet: 820 Cost: $151826 - Est: $153253 = Objective: $-1426.660 No
P3  Baths:1 Beds:1 Square feet: 710 Cost: $125943 - Est: $124516 = Objective: $ 1426.660 Selected
P4  Baths:3 Beds:4 Square feet:4300 Cost: $420801 - Est: $419374 = Objective: $ 1426.660 No
P5  Baths:2 Beds:4 Square feet:3800 Cost: $374751 - Est: $375784 = Objective: $-1032.637 No
P6  Baths:1 Beds:3 Square feet:2200 Cost: $251674 - Est: $250247 = Objective: $ 1426.660 No
P7  Baths:2 Beds:3 Square feet:3400 Cost: $332426 - Est: $331461 = Objective: $  965.237 No
```

4

GOAL

A broker has eleven loans of size ranging from $55,000 to $910,000. The broker would like to group the loans into packages so that each package has at least $1M in it, and the number of packages is maximized;

Object	Value ($1M)	Range	
A	910	$55.000	$910.000
B	870		
C	810		
D	640		
E	550		
F	250		
G	120		
H	95		
I	55		
J	543		
K	449		

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Business
- Bundling

Source:

- Book 4
- OBJBUNDL

```
MODEL:
SETS:
  OBJECT: VALUE, OVER, IND;
  OXO( OBJECT, OBJECT)| &1 #LE# &2: X;
ENDSETS
DATA:
  OBJECT =  A   B   C   D   E   F   G   H   I   J   K;
   VALUE = 910 870 810 640 550 250 120  95 55 543 449;
! The value in each bundle must be >= PKSIZE;
   PKSIZE = 1000;
ENDDATA
SUBMODEL MAX99:
!  This method may be time consuming for more  than 15 objects;
!  Definition of variables;
!         X( I, I) = 1 if object I is lowest numbered object in its package;
!         X( I, J) = 1 if object j is assigned to package I;
! Maximize number of packages assembled;
[OBJ] MAX = @SUM( OBJECT( I): X( I, I));
@FOR( OBJECT( K):
         ! Each object can be assigned to at most one package;
         @SUM( OXO( I, K): X( I, K)) <= 1;
         ! A package must be at least PSIZE in size;
         @SUM( OXO( K, J): VALUE( J) * X( K, J)) - OVER( K) = PKSIZE * X( K, K););
! The X( I, J) must = 0 or 1;
@FOR( OXO( I, J): @BIN( X( I, J)););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), '   Object   Value',@NEWLINE(1));
@FOR(OBJECT(I): @WRITE( ' ', @FORMAT(OBJECT(I),'6s'), '  ' ,
         @FORMAT(VALUE(I),'%7.0f'),
@NEWLINE(1)));
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MAX99);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  PACKAGE: ", @NEWLINE( 1));
@WRITEFOR(OXO(I,J)|X(J,J) #EQ# 0 #AND# X(I,J) #GT# 0: '  Package: ',@NAME(X(I,J)), ' ',
         @FORMAT(VALUE(I),'%4.0f'),' + ',
         @FORMAT(VALUE(J),'%3.0f'),' = ',
         @FORMAT(VALUE(I)+VALUE(J),'%4.0f'),' - 1000 = Over: ',
         @FORMAT(VALUE(I)+VALUE(J)-1000,'%3.0f'),'  ',
@NEWLINE(1));
@WRITE(' ',@NEWLINE(1));
@WRITE('  Note: ',@NEWLINE(1));
@WRITE('  As for the value of object B, the values of B+G+I were',@NEWLINE(1));
@WRITE('  added to meet the established premise of a minimum',@NEWLINE(1));
@WRITE('  value of 1000 for each package.',@NEWLINE(3));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX99);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 Object   Value
    A      910
    B      870
    C      810
    D      640
    E      550
    F      250
    G      120
    H       95
    I       55
    J      543
    K      449
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              5.000000
Objective bound:                              5.000000
Infeasibilities:                              0.000000
Total solver iterations:                            59

PACKAGE:
Package: X( A, H)  910 +  95 = 1005 - 1000 = Over:   5
Package: X( B, G)  870 + 120 =  990 - 1000 = Over: -10
Package: X( B, I)  870 +  55 =  925 - 1000 = Over: -75
Package: X( C, F)  810 + 250 = 1060 - 1000 = Over:  60
Package: X( D, K)  640 + 449 = 1089 - 1000 = Over:  89
Package: X( E, J)  550 + 543 = 1093 - 1000 = Over:  93

Note:
As for the value of object B, the values of B+G+I were
added to meet the established premise of a minimum
value of 1000 for each package.
```

BLOCK 4

Block: DIET

What types of food should be selected for school meal composition, obeying minimum nutrition criteria at the lowest cost?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Blend

Source:

- Book 11
- Slide 30

GOAL

Four types of food are available in the preparation of a group of children: chocolate biscuit, ice cream, soda and cheese pie.

The composition of these foods and their cost per serving are available below:

Formula		Biscuit	Ice Cream	Soda	Cheese Pie	Available
Calories	cal	400	200	150	500	500
Chocolate	g	3	2	0	0	6
Sugar	g	2	2	2	4	10
FAT	g	2	4	1	5	8
Cost	$	0.50	0.20	0.30	0.80	-

Children should eat at least 500 Calories, 6 g of Chocolate, 10 g of Sugar, and 8 g of FAT. Formulate the problem so that the cost is minimized

```
MODEL:
SETS:
 COMPOSITION :;
 RESOURCE: AVAILABLE,  COST, PRODUCE;
 RXP( COMPOSITION, RESOURCE) : FORMULA;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        COST,       AVAILABLE     =
 BISCUIT          0.50        500
 ICE_CREAM        0.20        6
 SODA             0.30        10
 CHEESE_PIE       0.8         8;
! Composition attributes;
 COMPOSITION  =
 CALORIES
 CHOCOLATE
 SUGAR
 FAT ;

! Required        Biscuit     Ice_cream   Soda        Cheese pie;
 FORMULA  =       400         200         150         500         ! Calories;
                  3           2           0            0          ! Chocolate;
                  2           2           2            4          ! Sugar;
                  2           4           1            5;         ! fat;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM( RESOURCE( p): COST( p) * PRODUCE( p));
! The available constraint;
@FOR( RESOURCE( r):
      [AVA] @SUM( COMPOSITION( p): FORMULA( r, p) * PRODUCE( p )) >= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (cal,g,g,g):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (cal,g,g,g):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST p/un:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RESOURCE( J)| PRODUCE( J) #GT# 0: '   ',
      @FORMAT(RESOURCE( J),'-10s'),
      @FORMAT(PRODUCE( J),'%6.0f'),'  un x unit cost: $',
      @FORMAT(COST( J), '%4.2f'),' = Total: $',
      @FORMAT(COST( J) * PRODUCE( J),'%4.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (cal,g,g,g)):
                BISCUIT   ICE_CREAM       SODA  CHEESE_PIE
 CALORIES      400.0000    200.0000   150.0000    500.0000
 CHOCOLATE     3.000000    2.000000   0.000000    0.000000
 SUGAR         2.000000    2.000000   2.000000    4.000000
 FAT           2.000000    4.000000   1.000000    5.000000

AVAILABLE (cal,g,g,g)):
 BISCUIT       500.0000
 ICE_CREAM     6.000000
 SODA          10.00000
 CHEESE_PIE    8.000000

COST p/un:
 BISCUIT       0.5000000
 ICE_CREAM     0.2000000
 SODA          0.3000000
 CHEESE_PIE    0.8000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
SOLUTION:
Global optimal solution found.
Objective value:                              1.000000
Infeasibilities:                              0.000000

IDEAL DIET PROGRAM:
ICE_CREAM       5 un x unit cost: $0.20 = Total: $1.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Blend
- Vitamins

Source:

- Book 2
- Page 226

GOAL

It is known that milk, meat and eggs provide the amounts of vitamins reported in the table below:

Formula		Vitamin A (mg)	Vitamin B (mg)	Vitamin C (mg)	Cost ($)
Milk	L	0.25	2.0	10.0	2.20
Meat	kg	25.0	20.0	10.0	17.00
Eggs	12un	2.5	200.0	10.0	4.20
Required	mg	1.0	50.0	19.0	

Calculate the amount of milk, meat and eggs that meets the basic daily needs of nutrients at a minimal cost

```
MODEL:
SETS:
 COMPOSITION : COST;
 RESOURCE: REQUIRED, PRODUCE;
 RXP( COMPOSITION, RESOURCE) : FORMULA;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        REQUIRED      =
 VIT_A            1
 VIT_B            50
 VIT_C            19;
! Composition attributes;
 COMPOSITION,     COST  =
 MILK             2.20
 CARNE            17.00
 OVOS             4.20;
! Formula in mg   MILK (L)     CARNE (kg)   OVOS (12);
 FORMULA  =       0.25         2            10          ! VIT_A;
                  25           20           10          ! VIT_B;
                  2.50         200          10;         ! VIT_C;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN = @SUM( RESOURCE( p): COST( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( COMPOSITION( p): FORMULA( r, p) * PRODUCE( p )) >= REQUIRED( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data  block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (mg)):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  REQUIRED (mg)):", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  COST ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( COMPOSITION( J)| PRODUCE( J) #GT# 0: '  ',
        @FORMAT(COMPOSITION( J),'-8s'),
        @FORMAT(PRODUCE( J),'%6.3f'),@IF(J #EQ# 1,' L ', @IF(J #EQ# 3, ' un',' kg')),'  x  Unit cost: $',
        @FORMAT(COST( J), '%5.2f'),' = Total: $',
        @FORMAT(COST( J) * PRODUCE( J),'%4.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (mg)):
            VIT_A     VIT_B     VIT_C
 MILK   0.250000  2.000000  10.00000
 BEEF   25.00000  20.00000  10.00000
 EGGS   2.500000  200.0000  10.00000

REQUIRED (mg)):
 VIT_A  1.000000
 VIT_B  50.00000
 VIT_C  19.00000

COST ($):
 MILK   2.200000
 BEEF   17.00000
 EGGS   4.200000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              5.577315
Infeasibilities:                              0.000000

IDEAL DIET PROGRAM:
MILK     1.930 L    x  Unit cost: $ 2.20 = Total: $4.25
BEEF     0.069 kg   x  Unit cost: $17.00 = Total: $1.17
EGGS     0.038 un   x  Unit cost: $ 4.20 = Total: $0.16
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Vitamins

GOAL

Suppose that, for justifiable reasons, a diet is restricted to skimmed milk, lean beef, fish meat and a salad of well-known composition.

Formula		Milk (L)	Meat (kg)	Fish (kg)	Salad (100 g)	Daily Need
Vitamin A	mg	2.0	2.0	20.0	20.0	11.0
Vitamin B	mg	50.0	20.0	10.0	30.0	70.0
Vitamin C	mg	80.0	70.0	10.0	80.0	250.0
Cost	$	2.00	4.00	1.50	1.00	-

It is also known that nutritional requirements are expressed in terms of vitamins A, C and D and controlled by their minimum amounts (in milligrams), since they are indispensable to the health of the person who are undergoing the diet.

In the details it boils down to the amount of each vitamin in food availability and its daily requirement for a person's good health.

Formulate the program to optimize the resources involved

```
MODEL:
SETS:
 PRODUCT :COST, PRODUCE;
 RESOURCE:NEED;
 RXP( RESOURCE, PRODUCT ) : FORMULA;
ENDSETS
DATA:
! Products attributes;
 PRODUCT,        COST  =
 MILK            2
 MEAT            4
 FISH            1.5
 SALAD           1;
! Resources attributes;
 RESOURCE,       NEED  =
 VIT_A           11
 VIT_B           70
 VIT_C           250;
! Required (mg)  Milk      Meat      Fish      Salad;
 FORMULA    =    2         2         10        20        ! VIT_A;
                 50        20        10        30        ! VIT_B;
                 80        70        10        80;       ! VIT_C;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! The Minimum Required Constraints;
@FOR( RESOURCE( r):
      @SUM( PRODUCT( p): FORMULA( r, p) * PRODUCE( p )) >= NEED( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (mg)):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  DAILY NEED (mg)):", @NEWLINE( 1));
@TABLE(NEED);
@WRITE(" ", @NEWLINE( 1), "  COST per (L,kg, kg,100g):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE( J) #GT# 0: '  ',
      @FORMAT(PRODUCT( J),'-8s'),
      @FORMAT(PRODUCE( J),'%5.3f'),' Grams x Unit cost: $',
      @FORMAT(COST( J), '%4.2f'), ' = Total: $',
      @FORMAT(COST( J) * produce( J),'%5.3f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (mg)):
              MILK      MEAT      FISH     SALAD
 VIT_A  2.000000  2.000000  10.00000  20.00000
 VIT_B  50.00000  20.00000  10.00000  30.00000
 VIT_C  80.00000  70.00000  10.00000  80.00000

DAILY NEED (mg)):
 VIT_A  11.00000
 VIT_B  70.00000
 VIT_C  250.0000

COST per (L,kg, kg,100g):
 MILK   2.000000
 MEAT   4.000000
 FISH   1.500000
 SALAD  1.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              3.125000
Infeasibilities:                              0.000000


IDEAL DIET PROGRAM:
SALAD   3.125 Grams x Unit cost: $1.00 = Total: $3.125
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Vitamins

Source:

- Book 11
- Ex.: 20

GOAL

The unit requirements of a pig fattening ration are those indicated in the MINREQ table, per kg of feed.

To achieve these specific values, up to 50% of the requirements of Methionine for Cystine can be substituted. In addition, a ratio of 1.5 to 2 : 1, and 1.5 to 2.0 parts of Calcium to 1 part of Phosphorus, must be obeyed for the amount of Calcium and Phosphorus.

The food used to make the feed, as well as its nutrient content and price are shown below:

Formula		Corn	Sorghum	Soy-Flour	Blood-Flour	Bone-Flour	Required
Protein	g	100.0	0.90	260.0	930.0	0	0.14
Methionine	g	10.0	13.0	20.0	10.6	0	2.2
Cis-Tina	g	1.5	1.6	6.5	11.5	0	2.2
Calcium	g	1.0	0.3	2.9	0.7	308.5	8.0
Phosphor	g	2.5	3.0	10.5	11.2	141.3	4.0
Cost p/kg	$	1.20	0.96	2.30	4.30	1.83	-

The objective is to determine the composition of ration that offers the minimum possible cost per kilo, taking into account the requirements placed in the formula.

Elaborate a model of Linear Programming to achieve this goal.

```
MODEL:
SETS:
 COMPOSITION: MINREQ ;
 RESOURCE: COST, PRODUCE;
 RXP( COMPOSITION, RESOURCE) : FORMULA;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        COST =
 CORN             1.20
 SORGHUM          0.96
 SOYFLOUR         2.30
 BLOODFLOUR       4.30
 BONEFLOUR        1.83;

! Composition attributes;
 COMPOSITION,     MINREQ =
 PROTEIN          0.14
 METHIONINE       2.2
 CISTINA          2.2
 CALCIUM          8.0     ! Ratio 2 : 1 or 2 parts Calcium from 1 part Phosphor;
 PHOSPHOR         4.0;

! Required        CORN     SORGHUM   SOYFLOUR   BLOODFLOUR    BONEFLOUR;
 FORMULA  =       100.0    0.90      260.0      930.0         0.00          ! PROTEIN (g/);
                  10.0     13.0      20.0       10.6          0.00          ! METHIONINE (g);
                  1.50     1.60      6.50       11.5          0.00          ! CISTINA (g);
                  1.00     0.30      2.90       0.70          308.5         ! CALCIUM (g);
                  2.50     3.00      10.5       11.2          141.3;        ! PHOSPHOR (g);
ENDDATA
SUBMODEL MIN4:
[OBJ] MIN = @SUM( RESOURCE( p): COST( p) * PRODUCE( p));
! The Minimum Required Constraints;
@FOR( RESOURCE( r):
      @SUM( COMPOSITION( p): FORMULA( r, p) * PRODUCE( p )) >= MINREQ( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (g):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM REQUIRED (g):", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN4);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RESOURCE( J)| PRODUCE( J) #GT# 0: '  ',
      @FORMAT(RESOURCE( J),'-10s'),'   ',
      @FORMAT(PRODUCE( J),'%4.2f'),' kg x Unit cost: $',
      @FORMAT(COST( J), '%4.2f'),' = Total: $',
      @FORMAT(COST( J) * PRODUCE( J),'%4.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (g):
                   CORN      SORGHUM     SOYFLOUR  BLOODFLOUR    BONEFLOUR
 PROTEIN       100.0000    0.9000000     260.0000    930.0000     0.000000
 METHIONINE    10.00000     13.00000     20.00000    10.60000     0.000000
 CISTINA       1.500000     1.600000     6.500000    11.50000     0.000000
 CALCIUM       1.000000    0.3000000     2.900000   0.7000000     308.5000
 PHOSPHOR      2.500000     3.000000     10.50000    11.20000     141.3000

MINIMUM REQUIRED (g):
 PROTEIN     0.1400000
 METHIONINE   2.200000
 CISTINA      2.200000
 CALCIUM      8.000000
 PHOSPHOR     4.000000

COST:
 CORN         1.200000
 SORGHUM      0.960000
 SOYFLOUR     2.300000
 BLOODFLOUR   4.300000
 BONEFLOUR    1.830000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              0.820095
Infeasibilities:                              0.000000

IDEAL DIET PROGRAM:
SOYFLOUR     0.34 kg x Unit cost: $2.30 = Total: $0.78
BONEFLOUR    0.02 kg x Unit cost: $1.83 = Total: $0.04
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Vitamins

GOAL

Suppose, for nutritional reasons, that a person has entered the Academy to lose weight and acquire a more defined bodybuilding, when consulting the nutritionalist, he received a diet for weight reduction and gain of lean mass. It intends to ingest the nutrients needed to maintain health by minimizing the number of calories. Consider the formula below, as it should be her diet.

Portion (100 g)	Protein (g)	Carbo (g)	Lipids (g)	Calories (kcal)
Rice	2.6	25.8	1.0	123.5
Eggs	13.3	0.6	9.5	145.7
Chicken	32.0	0	2.5	159.2
Meat	35.9	0	7.3	219.3
Manioc	0.6	30.1	0.2	119.0
Bean	4.8	13.6	0.5	76.4
Required (g)	100.0	100.0	11.0	-

```
MODEL:
SETS:
 COMPOSITION : MINREQ;
 RESOURCE: CALORIES, PRODUCE;
 RXP( RESOURCE, COMPOSITION) : FORMULA;
ENDSETS
DATA:
! Resources attributes (CALORIES FOR EACH PORTION OF 100 g);
 RESOURCE,        CALORIES    =
 RICE             123.5
 EGGS             145.7
 CHICKEN          159.2
 BEEF             219.3
 MANIOC           119.0
 BEAN             76.4;

! Composition attributes (g);
 COMPOSITION,     MINREQ      =
 PROTEIN          100
 CARBO            320
 LIPIDS           11;

! Required        PROTEIN     CARBO      LIPIDs;
 FORMULA  =       2.6         25.8       1          ! RICE;
                  13.3        0.6        9.5        ! EGGS;
                  32.0        0          2.5        ! CHICKEN;
                  35.9        0          7.3        ! BEEF;
                  0.6         30.1       0.20       ! MANIOC;
                  4.8         13.6       0.5;       ! BEAN;
ENDDATA
SUBMODEL MIN5:
[OBJ] MIN = @SUM( RESOURCE( p): CALORIES( p) * PRODUCE( p));
! The Minimum Required Constraints;
@FOR( COMPOSITION( COL ):
      @SUM( RESOURCE( LIN ): FORMULA( LIN, COL) * PRODUCE( LIN )) >= MINREQ( COL ););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (g)):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM REQUIRED (g):", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  CALORIES (kcal):", @NEWLINE( 1));
@TABLE(CALORIES);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITE("  REQUIRED:", @NEWLINE( 1));
@WRITEFOR( RESOURCE( J)| PRODUCE( J) #GT# 0: '  .',
      @FORMAT(RESOURCE( J),'-8s'), ' ',
      @FORMAT(PRODUCE( J),'%7.2f'),' Portions x',
      @FORMAT(CALORIES( J), '%6.1f'),' Kcal = ',
      @FORMAT(CALORIES( J) * PRODUCE( J),'%8.3f'),' Kcal',
@NEWLINE( 1));
@WRITE("  .Total ",34*'.',' ',@FORMAT(OBJ, '%8.3f'),' Kcal',  @NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (gr)):
              PROTEIN      CARBO     LIPIDS
 RICE        2.600000   25.80000   1.000000
 EGGS        13.30000   0.600000   9.500000
 CHICKEN     32.00000   0.000000   2.500000
 BEEF        35.90000   0.000000   7.300000
 MANIOC      0.600000   30.10000   0.200000
 BEAN        4.800000   13.60000   0.500000

MINIMUM REQUIRED (gr):
 PROTEIN     100.0000
 CARBO       320.0000
 LIPIDS      11.00000

CALORIES (kcal):
 RICE        123.5000
 EGGS        145.7000
 CHICKEN     159.2000
 BEEF        219.3000
 MANIOC      119.0000
 BEAN        76.40000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                          1736.739
Infeasibilities:                          0.000000

IDEAL DIET PROGRAM:
REQUIRED:
.EGGS        0.05 Portions x 145.7 Kcal =    7.129 Kcal
.MANIOC      1.35 Portions x 119.0 Kcal =  161.241 Kcal
.BEAN       20.53 Portions x  76.4 Kcal = 1568.369 Kcal
.Total ................................. 1736.739 Kcal
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Food
- Formula
- Vitamins

Source:

- Book 1
- Page 55

GOAL

To elaborate a model of Linear Programming, that satisfies the feeding needs of a determined animal. In the table below the specifications follow:

Formula	Protein (un/k)	Carbo (un/k)	Cost ($)
RA	25	55	3.00
RB	25	20	2.00
RC	45	10	4.00
RD	35	35	3.00
RE	25	20	3.00
Required (un/k)	200	250	-

```
MODEL:
SETS:
 COMPOSITION : MINREQ;
 RESOURCE: COST, PRODUCE;
 RXP( RESOURCE, COMPOSITION) : FORMULA;
ENDSETS
DATA:
! Resources attributes (COST /k);
 RESOURCE,       COST =
 RA              3.00
 RB              2.00
 RC              4.00
 RD              3.00
 RE              3.00;

! Composition attributes (k);
 COMPOSITION,    MINREQ     =
 PROTEIN         200
 CARBO           250;

! Required    (k)    PROTEIN    CARBO;
 FORMULA  =          25         55          ! RA;
                     25         20          ! RB;
                     45         10          ! RC;
                     35         35          ! RD;
                     25         20;         ! RE;
ENDDATA
SUBMODEL MIN6:
[OBJ] MIN = @SUM( RESOURCE( p): COST( p) * PRODUCE( p));
! The Minimum Required Constraints;
@FOR( COMPOSITION( COL ):
      [REQ] @SUM( RESOURCE( LIN ): FORMULA( LIN, COL) * PRODUCE( LIN )) >= MINREQ( COL );); 
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (un/k):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM REQUIRED (un/k):", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  COST per/k:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN6);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL DIET PROGRAM: ", @NEWLINE( 1));
@WRITE("  REQUIRED:", @NEWLINE( 1));
@WRITEFOR( RESOURCE( J)| PRODUCE( J) #GT# 0: '  .',
@FORMAT(RESOURCE( J),'-5s'),
@FORMAT(PRODUCE( J),'%4.2f'),' k   x   Unit Cost: $',
@FORMAT(COST( J), '%4.2f'),'   =  Total: $',
@FORMAT(COST( J) * PRODUCE( J),'%5.2f'),  @NEWLINE( 1));
@WRITE("  Minimum cost: ",32*' ',' $',@FORMAT(OBJ, '%5.2f'),  @NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (un/k):
      PROTEIN     CARBO
 RA  25.00000  55.00000
 RB  25.00000  20.00000
 RC  45.00000  10.00000
 RD  35.00000  35.00000
 RE  25.00000  20.00000

MINIMUM REQUIRED (un/k):
 PROTEIN  200.0000
 CARBO    250.0000

COST per/k:
 RA  3.000000
 RB  2.000000
 RC  4.000000
 RD  3.000000
 RE  3.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              18.57143
Infeasibilities:                              0.000000

IDEAL DIET PROGRAM:
REQUIRED:
.RA   2.57 k   x   Unit Cost: $3.00   =  Total: $ 7.71
.RB   5.43 k   x   Unit Cost: $2.00   =  Total: $10.86
Minimum cost:                                   $18.57
```

BLOCK 5

Block: AVIATION

What is the best distribution of cargo or the best alternative fuel supply per airport or better distribution of seats by class and by aircraft, so as to minimize costs and maximize profit?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Load
- Transport

Source:

- Book 3
- Chapter 2.4.25

GOAL

A cargo plane has four cargo compartments: Front, Center, Tail and Basement.The first three can only receive cargo in containers, while the hold receives bulk material.

In order to balance the flight, it is essential that the load distribution be proportional between the compartments.

There are 3 types of containers and two types of bulk cargoes to load the aircraft. The 2 types of bulk cargo can be transported together.

The cargo distribution must meet the capacity constraints of the airplane. The information is detailed below:

Compartments			Front	Center	Tail	Bulk Dump	
Cargo	Volume	m3	0.50	1.00	0.25	1.00	
	Weight	ton	1.10	1.80	0.25	1.2	1.7
	Container	ton	0.35	0.90	0.05		
Aircraft	Capacity	ton	7	7	6	3.5	3.5
	Volume	m3	35	55	30	30	
Profit		$	200.00	220.00	175.00	235.00	180.00

To elaborate the problem of the linear programming that optimizes the distribution of the load in order to maximize the profit of the flight of the freighter.

```
MODEL:
SETS:
HEADER1 / PROD, LIMIT, VALUE /:;
RESOURCE: CAR_VOL, CAR_WEI, PROFIT, AIR_CAP, AIR_VOL, PRODUCE;
PXR (RESOURCE,HEADER1)  : SLASUR1;
ENDSETS
DATA:
! Resources attributes;
RESOURCE ,  CAR_VOL,    CAR_WEI,    PROFIT,     AIR_CAP,    AIR_VOL     =
FRONT       0.5         1.1         200         7           35
CENTER      1           1.8         220         7           55
TAIL        0.25        0.25        175         6           30
BULK1       1           1.2         235         3.5         15
BULK2       1           1.7         180         3.5         15;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX = @SUM( RESOURCE( J): PROFIT( J) * PRODUCE( J));
! Load weight restrictions;
@FOR(RESOURCE(K): [CARWEI]  PRODUCE(K) <= CAR_WEI(K));
! Weight restrictions by compartment;
@FOR(RESOURCE(C): [AIRCAP] PRODUCE(C) <= AIR_CAP(C));
! Restriction of space by compartment;
@FOR(RESOURCE(J): [AIRVOL] CAR_VOL(J) * PRODUCE(J) <= AIR_VOL(J));
!The equilibrium was not calculated, because for each type of aircraft, its weight distribution is predetermined by the manufacturer.;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  PROFIT BY TON ($):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  AIRCRAFT LOAD CAPACITY (Ton):", @NEWLINE( 1));
@TABLE(AIR_CAP);
@WRITE(" ", @NEWLINE( 1), "  AIRCRAFT LOAD CAPACITY (Ton):", @NEWLINE( 1));
@TABLE(AIR_VOL);
@WRITE(" ", @NEWLINE( 1), "  VOLUME OF LOAD (m3):", @NEWLINE( 1));
@TABLE(CAR_VOL);
@WRITE(" ", @NEWLINE( 1), "  LOAD WEIGHT (Ton):", @NEWLINE( 1));
@TABLE(CAR_WEI);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL SUPPLY PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RESOURCE(J) : '  Load in: ',
        @FORMAT(RESOURCE(J),'-6s'), '  ',
        @FORMAT(PRODUCE(J),'%4.2f'), ' ton x unit profit: $',
        @FORMAT(PROFIT(J),'%4.2f'), ' = Total: $',
        @FORMAT(PRODUCE(J) * PROFIT(J),'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
PROFIT BY TON ($):
 FRONT    200.0000
 CENTER   220.0000
 TAIL     175.0000
 BULK1    235.0000
 BULK2    180.0000

AIRCRAFT LOAD CAPACITY (Ton):
 FRONT    7.000000
 CENTER   7.000000
 TAIL     6.000000
 BULK1    3.500000
 BULK2    3.500000

AIRCRAFT LOAD CAPACITY (Ton):
 FRONT    35.00000
 CENTER   55.00000
 TAIL     30.00000
 BULK1    15.00000
 BULK2    15.00000
```

```
VOLUME OF LOAD (m3):
 FRONT    0.500000
 CENTER   1.000000
 TAIL     0.250000
 BULK1    1.000000
 BULK2    1.000000

LOAD WEIGHT (Ton):
 FRONT    1.100000
 CENTER   1.800000
 TAIL     0.250000
 BULK1    1.200000
 BULK2    1.700000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                         1247.750
Infeasibilities:                                         0.000000

IDEAL SUPPLY PROGRAM:
Load in: FRONT   1.10 ton x unit profit: $200.00 = Total: $220.00
Load in: CENTER  1.80 ton x unit profit: $220.00 = Total: $396.00
Load in: TAIL    0.25 ton x unit profit: $175.00 = Total: $ 43.75
Load in: BULK1   1.20 ton x unit profit: $235.00 = Total: $282.00
Load in: BULK2   1.70 ton x unit profit: $180.00 = Total: $306.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Fuel
- Purchase

Source:

- Book 1
- Page 193

GOAL

An aviation company wants to buy fuel for its planes, which can be supplied at 5 Airports served by it and in which the needs are 100, 200, 500, 800 and 100 thousand gallons.

The maximum fuel availability available in the four supplying companies in gallons is 1000, 1300, 400 and 1000 thousand.

Airport / Supplier		S1	S2	S3	S4	Demand (gal)
A	$	1.10	1.20	1.15	1.20	100,000
B	$	1.25	1.20	1.22	1.30	200,000
C	$	1.30	1.25	1.27	1.25	500,000
D	$	1.20	1.21	1.21	1.20	800,000
E	$	1.30	1.20	1.25	1.30	100,000
Capacity	gal	1,000,000	1,300,000	400,000	1,000,000	-

Are the fuel delivery cost detailed below, thereby determining the most economical purchasing scheme?

```
MODEL:
SETS:
SUPPLIER: CAPACITY;
CONSUMER: DEMAND;
ROUTES( CONSUMER, SUPPLIER): COST, VOLUME;
ENDSETS
DATA:
! Capacity attributes (gal);
SUPPLIER,            CAPACITY   =
SUPPLIER_1           1000000
SUPPLIER_2           1300000
SUPPLIER_3           4000000
SUPPLIER_4           1000000;
! Consumer attributes (gal);
CONSUMER,            DEMAND   =
AIRPORT_A            100000
AIRPORT_B            200000
AIRPORT_C            500000
AIRPORT_D            800000
AIRPORT_E            100000;
! Cost required       SUP1  SUP2  SUP3  SUP4;
COST          =      1.10  1.20  1.15  1.20              ! Airport A;
                     1.25  1.20  1.22  1.30              ! Airport B;
                     1.30  1.25  1.27  1.25              ! Airport C;
                     1.20  1.21  1.21  1.20              ! Airport D;
                     1.30  1.20  1.25  1.30;             ! Airport E;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! Restrictions to ensure that the demand for each airport will be carried out;
@FOR(CONSUMER(I):
        [DEM] @SUM(SUPPLIER(J): VOLUME(I,J)) = DEMAND(I););
! Restrictions to ensure the capacity of each supplier;
@FOR(SUPPLIER(J):
         [CAP] @SUM(CONSUMER(I): VOLUME( I, J) + VOLUME( I, J)) <= CAPACITY(J););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  COST FOR GAL ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SUPPLY CAPABILITY (gal):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (gal):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL SUPPLY PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
        @FORMAT(CONSUMER(I),'-9s'), ', ',
        @FORMAT(SUPPLIER(J),'-8s'), ', Provides:',
        @FORMAT(VOLUME( I, J),'%6.0f'),' Gal', ' x Unit profit:$',
        @FORMAT(COST(I,J),'%4.2f'), ' = Total:$',
        @FORMAT(COST(I, J) * VOLUME(I,J),'%9.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COST FOR GAL ($):
            SUPPLIER_1  SUPPLIER_2  SUPPLIER_3  SUPPLIER_4
 AIRPORT_A    1.100000    1.200000    1.150000    1.200000
 AIRPORT_B    1.250000    1.200000    1.220000    1.300000
 AIRPORT_C    1.300000    1.250000    1.270000    1.250000
 AIRPORT_D    1.200000    1.210000    1.210000    1.200000
 AIRPORT_E    1.300000    1.200000    1.250000    1.300000

SUPPLY CAPABILITY (gal):
 SUPPLIER_1  1000000.
 SUPPLIER_2  1300000.
 SUPPLIER_3  400000.0
 SUPPLIER_4  1000000.

DEMAND (gal):
 AIRPORT_A  100000.0
 AIRPORT_B  200000.0
 AIRPORT_C  500000.0
 AIRPORT_D  800000.0
 AIRPORT_E  100000.0
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                   2055500.
Infeasibilities:                                                   0.000000

IDEAL SUPPLY PROGRAM:
AIRPORT_A, SUPPLIER_1, Provides:100000 Gal x Unit profit:$1.10 = Total:$110000.00
AIRPORT_B, SUPPLIER_2, Provides:200000 Gal x Unit profit:$1.20 = Total:$240000.00
AIRPORT_C, SUPPLIER_2, Provides:350000 Gal x Unit profit:$1.25 = Total:$437500.00
AIRPORT_C, SUPPLIER_4, Provides:150000 Gal x Unit profit:$1.25 = Total:$187500.00
AIRPORT_D, SUPPLIER_1, Provides:400000 Gal x Unit profit:$1.20 = Total:$480000.00
AIRPORT_D, SUPPLIER_3, Provides: 50000 Gal x Unit profit:$1.21 = Total:$ 60500.00
AIRPORT_D, SUPPLIER_4, Provides:350000 Gal x Unit profit:$1.20 = Total:$420000.00
AIRPORT_E, SUPPLIER_2, Provides:100000 Gal x Unit profit:$1.20 = Total:$120000.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Transport
- Load

Source:

- Book 1
- Page 196

GOAL

A freight agent wants to set the next day's minimum cost schedule for cities A, B through aircraft A1, A2 and A3.

The information below shows the maximum load of each aircraft, as well as the respective cost, demands and limits.

resources / Aircraft				A1	A2	A3	Demand
Containers	Qty	Maximum	un	5	5	10	
		Available	un	10	15	20	
Cost	City	A	$	20.00	25.00	35.00	100 un
		B	$	25.00	30.00	33.00	150 un

```
MODEL:
SETS:
PRODUCTS: AVAILABLE, NMAX_CONT;
RESOURCES: DEMAND;
ROUTES( RESOURCES, PRODUCTS): COST, VOLUME;
ENDSETS
DATA:
! Resource attributes;
RESOURCES,          DEMAND   =
CITY_A              100
CITY_B              500;
! Products attributes;
PRODUCTS,           AVAILABLE,  NMAX_CONT   =
AIRCRA_1            10          5
AIRCRA_2            15          7
AIRCRA_3            20          10;

! Required          AIRCRA_1    AIRCRA_2    AIRCRA_3;
COST   =            20          25          35          ! cost per container per aircraft for city A;
                    25          30          33;         ! cost per container per aircraft for city B;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! Restrictions of containers demand for city A and B;
@FOR(RESOURCES(I):
        [DEM] @SUM(PRODUCTS(J): VOLUME(I,J)) <= DEMAND(I););
! Maximum container for aircraft;
@FOR(PRODUCTS(J):
         [NMC] @SUM(RESOURCES(I): VOLUME(I,J)) >= NMAX_CONT(J););
! Available container for aircraft;
@FOR(PRODUCTS(J):
        [AVA]  @SUM(RESOURCES(I): VOLUME(I,J)) <= AVAILABLE(J););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  COST OF FREIGHT FOR CONTAINER:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  MAXIMUM CONTAINERS LOAD PER AIRCRAFT (un):", @NEWLINE( 1));
@TABLE(NMAX_CONT);
@WRITE(" ", @NEWLINE( 1), "  CONTAINERS AVAILABLE (un):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN3)
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORTATION PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
        @FORMAT(RESOURCES(I),'-6s'), ', ',
        @FORMAT(PRODUCTS(J),'-8s'), ' Carries:',
        @FORMAT(VOLUME( I, J),'%3.0f'),' Containers', ' x Unit cost: $',
        @FORMAT(COST(I,J),'%4.2f'), ' = Total: $',
        @FORMAT(COST(I, J) * VOLUME(I,J),'%5.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COST OF FREIGHT FOR CONTAINER:
            AIRCRA_1  AIRCRA_2  AIRCRA_3
 CITY_A     20.00000  25.00000  35.00000
 CITY_B     25.00000  30.00000  33.00000

MAXIMUM CONTAINERS LOAD PER AIRCRAFT (un):
 AIRCRA_1   5.000000
 AIRCRA_2   7.000000
 AIRCRA_3   10.00000

CONTAINERS AVAILABLE (un):
 AIRCRA_1  10.00000
 AIRCRA_2  15.00000
 AIRCRA_3  20.00000

DEMAND (un):
 CITY_A    100.0000
 CITY_B    500.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                605.0000
Infeasibilities:                                                0.000000

IDEAL TRANSPORTATION PROGRAM:
CITY_A, AIRCRA_1 Carries:  5 Containers x Unit cost: $20.00 = Total: $100.00
CITY_A, AIRCRA_2 Carries:  7 Containers x Unit cost: $25.00 = Total: $175.00
CITY_B, AIRCRA_3 Carries: 10 Containers x Unit cost: $33.00 = Total: $330.00
```

4

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Planning
- Operation
- Control
- Air Traffic

Source:
- Book 4
- AirTrafficFloGrnDlay

GOAL
Determine when each of a set of flights should take off. Each flight is described by a desired/scheduled take off time, and the time intervals at which it will use various resources (take off runway, flight control sectors, landing runway).

Each resource (runway or flight control sector) is described by how many aircraft can simultaneously use it.

The objective is to minimize the total ground delay suffered by all flights, subject to no resource being over-utilized at any point in time.

The flights to be scheduled:
FLIGHT:

ORDDAL1	O'Hare to Dallas
DCADEN1	WashingtonDC to Denver
BOSABQ1	Boston to Albuquerque

Their scheduled start times;
FLIGHT:

ORDDAL1	10
DCADEN1	0
BOSABQ1	25

Description of flight, resource, start, end of usage of flight from Chicago to Dallas

FLIGHT	SECTOR	TMENT	TMOUT
ORDDAL1	ORDRNWY	0	5
ORDDAL1	SECTORN	5	20
ORDDAL1	SECTORC	20	110
ORDDAL1	SECTORS	110	135
ORDDAL1	DFWRNWY	135	140

of flight from Washington National to Denver

FLIGHT	SECTOR	TMENT	TMOUT
DCADEN1	DCARNWY	0	5
DCADEN1	SECTORE	5	60
DCADEN1	SECTORC	60	140
DCADEN1	SECTORW	140	245
DCADEN1	DENRNWY	245	250

of flight from Boston to Albuquerque;

FLIGHT	SECTOR	TMENT	TMOUT
BOSABQ1	BOSRNWY	0	5
BOSABQ1	SECTORNE	5	60
BOSABQ1	SECTORC	60	140
BOSABQ1	SECTORSW	140	355
BOSABQ1	ABQRNWY	355	360 ;

For this data set, if:

- ORDDAL1 departs on time it is in SECTORC from 30 to 120,
- DCADEN1 departs on time it is in SECTORC from 60 to 140,
- BOSABQ1 departs on time it is in SECTORC from 85 to 165,
-

So there would be 3 planes in SECTORC from 85 to 120, exceeding the capacity of 2, so one must be delayed.

```
MODEL:
! Air Traffic Flow Model with Ground Delays;

SETS:
!  Declare the various sets and the attributes associate with each member of the set;
 sector: scap;
 flight: tmbgn;
 time;
 sxt( sector, time): overload;
 fxt( flight, time): z;
 fxs( flight, sector): TMENT, TMOUT;
ENDSETS

DATA:
! Assume all times are in minutes;
! Minutes in a discrete time period/time bucket.the larger the bucket, the smaller the model and
 faster it solves. A smaller bucket providesa more accurate model;
 tmbkt = 5;
! Number discrete time periods in plan horizon;
 time = 1..80;
! Maximum delay tolerated for any flight;
 delaymx = 45;
! Weight in objective on total delay;
 delaywgt = 1;
! Wgt in objective on total overcongestion;
 congestwgt = 100;
!  The possible bottlenecks in the system. the runways, and the various air control sectors;
 sector = ORDRNWY  DCARNWY  DFWRNWY  DENRNWY BOSRNWY  ABQRNWY SECTORN
          SECTORNE  SECTORE  SECTORS  SECTORSW  SECTORW  SECTORC;
! Number planes/tasks that each resource can simultaneously handle.
   Each runway can handle only 1 plane at a time,
   Each sector can handle two simultaneously;
 scap =     1     1     1     1     1     1      2     2     2     2     2     2     2;
! The flights to be scheduled, O'Hare to Dallas, WashingtonDC to Denver, Boston to Albuquerque;
 flight = ORDDAL1  DCADEN1  BOSABQ1;
! Their scheduled start times;
 tmbgn =    10      0      25;
! Description of flight, resource, start, end of usage...;
  fxs,                        tment,          tmout =
!   of flight from Chicago to Dallas;
 ORDDAL1    ORDRNWY           0               5
 ORDDAL1    SECTORN           5              20
 ORDDAL1    SECTORC          20             110
 ORDDAL1    SECTORS         110             135
 ORDDAL1    DFWRNWY         135             140
!   of flight from Washington National to Denver;
 DCADEN1    DCARNWY           0               5
 DCADEN1    SECTORE           5              60
 DCADEN1    SECTORC          60             140
 DCADEN1    SECTORW         140             245
 DCADEN1    DENRNWY         245             250
!   of flight from Boston to Albuquerque;
 BOSABQ1    BOSRNWY           0               5
 BOSABQ1    SECTORNE          5              60
 BOSABQ1    SECTORC          60             140
 BOSABQ1    SECTORSW        140             355
 BOSABQ1    ABQRNWY         355             360 ;
! For this data set, if:
ORDDAL1 departs on time it is in SECTORC from 30 to 120,
DCADEN1 departs on time it is in SECTORC from 60 to 140,
BOSABQ1 departs on time it is in SECTORC from 85 to 165,
so there would be 3 planes in SECTORC from 85 to 120, exceeding the capacity of 2, so one must be delayed;
ENDDATA
```

```
SUBMODEL groundlay:
! Parameters:
   scap(s)       = number planes allowed in sector s simultaneously,
   tmbgn( f)     = scheduled begin time of flight f,
   tment( f, s)  = time flight f enters sector s minus depart time,
   tmout( f, s)  = time flight f exits sector s minus depart time;
! Variables:  z(f,b) = 1 if flight f takes off at time bucket b ;
! Bucket b begins at tmbkt*(b-1) and ends at tmbkt*b;
! An activity starting at p and ending at q overlaps bucket b if p < b*tmbkt, and q > tmbkt*(b-1),
  We say a flight f is delayed if it does not start in its earliest possible bucket, i.e.,
  bucket b so (b-1)*tmbkt >= tmbgn(f);

! Objective: Minimize the cost of ground delay + congestion;
 Min =          delaywgt * delaytot + congestwgt * overloadtot;
                overloadtot = @sum( sxt(s,b): overload(s,b));
                delaytot    = @sum( fxt(f,b): tmbkt*(b-1 - @floor(tmbgn(f)/tmbkt))*z(f,b));
! Each flight f must depart in some time bucket b;
 @for( flight(f):
   [MUSTDO] @sum( time(b): z(f,b)) = 1;);
! Number aircraft in sector s in time period t <= capacity + tolerated_congestion;
  @for( sector( s):
   @for( time(b):
    ! Sum over departure times t1 that would put plane in this sector at time bucket b;
     [SCTCAP] @sum( fxs(f,s):
          @sum( time(b1) | (b1 + tment(f,s)/tmbkt #le# b) #and# (b1 + tmout(f,s)/tmbkt #gt# b):
            z(f,b1))) <= scap(s) + overload(s,b););
  );
! The z(f,b) must be binary/(0 or 1);
  @for( fxt(f,b): @bin( z(f,b)));
! Set to zero infeasible departure times;
 @for( fxt( f,b) | (b-1)*tmbkt #lt# tmbgn(f) #or# (b-1)*tmbkt #gt# tmbgn(f) + delaymx:
        z(f,b) = 0;);
! Possible extensions:
   Flight connections:     Flight f2 cannot depart any earlier than x minutes after arrival of flight f1,
   Alternate flight paths: There are two or more alternate paths for a flight, exactly one must be taken.;
ENDSUBMODEL
CALC:
! Output level (0:verb, 1:terse, 2:only errors, 3:none);
 @SET( 'TERSEO',1);
! Set ending relative optimality tolerance;
 @SET( 'IPTOLR', .02);
! Time in seconds to apply optimality tolerance;
 @SET( 'TIM2RL', 5);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Solve the model;
 @SOLVE( groundlay);
! Solution report;
 @WRITE(" ", @NEWLINE( 1));
 @WRITE('  Scheduling of Ground Delays for a Set of Flights',@NEWLINE(2));
 @WRITE('    Flight     Sched_depart     Actual      Total Delay',@NEWLINE(1));
 @FOR( flight(f):
   dtime = @sum( fxt(f,b): tmbkt*(b-1)*z(f,b));
   @WRITE( ' ',@format( flight(f),'10s'),'     ',
   @FORMAT( tmbgn(f),'7.0f'),'       ',
   @FORMAT( dtime,'7.0f'),'       ',
   @FORMAT( dtime - tmbgn(f) ,'7.0f'), ' min',@NEWLINE(1))
 );
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

The flights to be scheduled:
FLIGHT:

ORDDAL1	O'Hare to Dallas
DCADEN1	WashingtonDC to Denver
BOSABQ1	Boston to Albuquerque

Their scheduled start times;
FLIGHT:

ORDDAL1	10
DCADEN1	0
BOSABQ1	25

Description of flight, resource, start, end of usage of flight from Chicago to Dallas

FLIGHT	SECTOR	TMENT	TMOUT
ORDDAL1	ORDRNWY	0	5
ORDDAL1	SECTORN	5	20
ORDDAL1	SECTORC	20	110
ORDDAL1	SECTORS	110	135
ORDDAL1	DFWRNWY	135	140

of flight from Washington National to Denver

FLIGHT	SECTOR	TMENT	TMOUT
DCADEN1	DCARNWY	0	5
DCADEN1	SECTORE	5	60
DCADEN1	SECTORC	60	140
DCADEN1	SECTORW	140	245
DCADEN1	DENRNWY	245	250

of flight from Boston to Albuquerque;

FLIGHT	SECTOR	TMENT	TMOUT
BOSABQ1	BOSRNWY	0	5
BOSABQ1	SECTORNE	5	60
BOSABQ1	SECTORC	60	140
BOSABQ1	SECTORSW	140	355
BOSABQ1	ABQRNWY	355	360 ;

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
Global optimal solution found.
Objective value:                              35.00000
Objective bound:                              35.00000
Infeasibilities:                              0.000000
Extended solver steps:                               0
Total solver iterations:                             0
Elapsed runtime seconds:                          0.68

Scheduling of Ground Delays for a Set of Flights

  Flight      Sched_depart      Actual      Total Delay
  ORDDAL1          10             10            0 min
  DCADEN1           0              0            0 min
  BOSABQ1          25             60           35 min
```

5

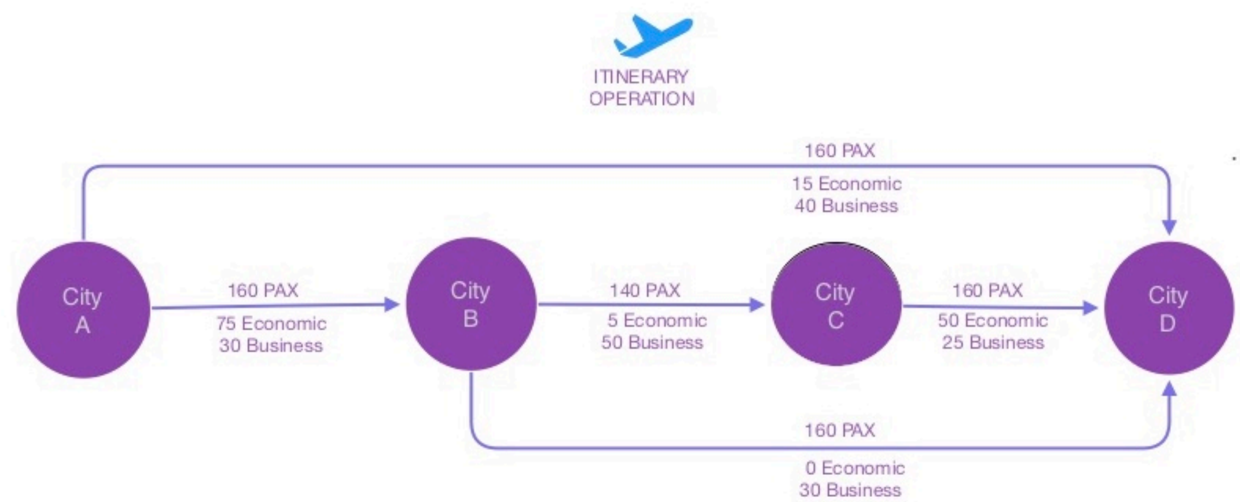
Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Planning
- Transport
- Operation

GOAL

A regional air transport company serves the following cities: A, B, C and D, which generate six possible routes shown in the data below.

Except A-B and C-D are operated with 160-seat aircraft (type 1) divided into two classes (executive and economy), while B-C is covered by a 140-seat aircraft (type 2), which also has two similar class options.



Resources / Routes			A-B		B-C		C-D		A-C		A-D		B-D	
Class	Demand	pax	90	30	40	50	50	25	70	30	80	40	60	30
	Category		Y	C	Y	C	Y	C	Y	C	Y	C	Y	C
Capacity / Aircraft	T1	pax	160		-		160		160		160		160	
	T2	pax	-		140		-		140		140		140	
Profit		$	300	400	200	250	300	360	400	480	700	840	400	450

Considering that the call demand estimates were obtained through the application of a directional traffic model, and the type of aircraft fleet planning, and that the company does not practice overbooking, a strategy that involves the sale of a larger number of seats than available, the effects of cancellations, formulate a model of allocation of fleet to maximize the revenue of this network.

```
MODEL:
SETS:
PRODUCT: PROFIT_EXEC, PROFIT_ECON, DEM_EXEC, DEM_ECON, AIRCRAFT_T1, AIRCRAFT_T2, PROD_EXEC,  PROD_ECON;
ENDSETS
DATA:
! Products attributes;
PRODUCT,    PROFIT_EXEC,        PROFIT_ECON,        DEM_EXEC,   DEM_ECON,   AIRCRAFT_T1, AIRCRAFT_T2   =
A_TO_B      300                 400                 90          30          160          0
B_TO_C      200                 250                 40          50          0            140
C_TO_D      300                 360                 50          25          160          0
A_TO_C      400                 480                 70          30          160          140
A_TO_D      700                 840                 80          40          160          140
B_TO_D      400                 450                 60          30          160          140;
ENDDATA
SUBMODEL MAX5:
[OBJ] MAX =  @SUM( PRODUCT(J): PROFIT_EXEC(J) * PROD_EXEC( J)) +
             @SUM( PRODUCT(J): PROFIT_ECON(J) * PROD_ECON( J)) ;
!  Demand restriction for business class;
@FOR(PRODUCT(J):
        [DEM_ECO] PROD_EXEC(J) <= DEM_EXEC(J));
@FOR(PRODUCT(J):
        [DEM_EXE] PROD_ECON(J) <= DEM_ECON(J));
! 160 PAX AIRCRAFT ROUTES (A-B, A-C, A-D);
[A160A]
PROD_EXEC(1) + PROD_ECON(1) +
PROD_EXEC(4) + PROD_ECON(4) +
PROD_EXEC(5) + PROD_ECON(5) <= 160;
! 140 PAX AIRCRAFT ROUTES (B-C, A-C, A-D, B-D);
[A140]
PROD_EXEC(2) + PROD_ECON(2) +
PROD_EXEC(4) + PROD_ECON(4) +
PROD_EXEC(5) + PROD_ECON(5) +
PROD_EXEC(6) + PROD_ECON(6) <= 140;
! 160 PAX AIRCRAFT ROUTES (C-D, A-D, B-D);
[A160B]
PROD_EXEC(3) + PROD_ECON(3) +
PROD_EXEC(5) + PROD_ECON(5) +
PROD_EXEC(6) + PROD_ECON(6) <= 160;
ENDSUBMODEL;
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  PROFIT EXECUTIVE CLASS P/PAX:", @NEWLINE( 1));
@TABLE(PROFIT_EXEC);
@WRITE(" ", @NEWLINE( 1), "  PROFIT ECONOMIC CLASS P/PAX:", @NEWLINE( 1));
@TABLE(PROFIT_ECON);
@WRITE(" ", @NEWLINE( 1), "  DEMAND EXECUTIVE CLASS P/PAX:", @NEWLINE( 1));
@TABLE(DEM_EXEC);
@WRITE(" ", @NEWLINE( 1), "  DEMAND ECONOMIC CLASS P/PAX:", @NEWLINE( 1));
@TABLE(DEM_ECON);
@WRITE(" ", @NEWLINE( 1), "  ROUTES OPERATED WITH AIRCRAFT T1 (160PAX):", @NEWLINE( 1));
@TABLE(AIRCRAFT_T1);
@WRITE(" ", @NEWLINE( 1), "  ROUTES OPERATED WITH AIRCRAFT T2 (140PAX):", @NEWLINE( 1));
@TABLE(AIRCRAFT_T2);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX5);
! Solution Report;
@WRITE(" EXECUTIVE CLASS:", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PROD_EXEC(J) #GT# 0: '  Route: ',
        @FORMAT(PRODUCT(J),'-8s'), '  #pax:',
        @FORMAT(PROD_EXEC(J),'%3.0F'), ' x  Profit p/pax: ' ,
        @FORMAT(PROFIT_EXEC(J),'%3.0f'), ' = Total:$ ',
        @FORMAT(PROFIT_EXEC(J) * PROD_EXEC(J),'%6.0f'),
@NEWLINE( 1));
@WRITE(" ECONOMIC CLASS:", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PROD_ECON(J) #GT# 0: '  Route: ',
        @FORMAT(PRODUCT(J),'-8s'), '  #pax:',
        @FORMAT(PROD_ECON(J),'%3.0F'), ' x  Profit p/pax: ' ,
        @FORMAT(PROFIT_ECON(J),'%3.0f'), ' = Total:$ ',
        @FORMAT(PROFIT_ECON(J) * PROD_ECON(J),'%6.0f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
PROFIT EXECUTIVE CLASS P/PAX:
 A_TO_B  300.0000
 B_TO_C  200.0000
 C_TO_D  300.0000
 A_TO_C  400.0000
 A_TO_D  700.0000
 B_TO_D  400.0000

PROFIT ECONOMIC CLASS P/PAX:
 A_TO_B  400.0000
 B_TO_C  250.0000
 C_TO_D  360.0000
 A_TO_C  480.0000
 A_TO_D  840.0000
 B_TO_D  450.0000

DEMAND EXECUTIVE CLASS P/PAX:
 A_TO_B  90.00000
 B_TO_C  40.00000
 C_TO_D  50.00000
 A_TO_C  70.00000
 A_TO_D  80.00000
 B_TO_D  60.00000

DEMAND ECONOMIC CLASS P/PAX:
 A_TO_B  30.00000
 B_TO_C  50.00000
 C_TO_D  25.00000
 A_TO_C  30.00000
 A_TO_D  40.00000
 B_TO_D  30.00000

ROUTES OPERATED WITH AIRCRAFT T1 (160PAX):
 A_TO_B  160.0000
 B_TO_C  0.000000
 C_TO_D  160.0000
 A_TO_C  160.0000
 A_TO_D  160.0000
 B_TO_D  160.0000

ROUTES OPERATED WITH AIRCRAFT T2 (140PAX):
 A_TO_B  0.000000
 B_TO_C  140.0000
 C_TO_D  0.000000
 A_TO_C  140.0000
 A_TO_D  140.0000
 B_TO_D  140.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
 Global optimal solution found.
 Objective value:                                     129600.0
 Infeasibilities:                                     0.000000

EXECUTIVE CLASS:
 Route: A_TO_B    #pax: 75 x  Profit p/pax: 300 = Total:$  22500
 Route: B_TO_C    #pax:  5 x  Profit p/pax: 200 = Total:$   1000
 Route: C_TO_D    #pax: 50 x  Profit p/pax: 300 = Total:$  15000
 Route: A_TO_D    #pax: 15 x  Profit p/pax: 700 = Total:$  10500
ECONOMIC CLASS:
 Route: A_TO_B    #pax: 30 x  Profit p/pax: 400 = Total:$  12000
 Route: B_TO_C    #pax: 50 x  Profit p/pax: 250 = Total:$  12500
 Route: C_TO_D    #pax: 25 x  Profit p/pax: 360 = Total:$   9000
 Route: A_TO_D    #pax: 40 x  Profit p/pax: 840 = Total:$  33600
 Route: B_TO_D    #pax: 30 x  Profit p/pax: 450 = Total:$  13500
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Purchase

Source:

- Book 2
- Chapter 2.5.9

GOAL

An airline wants to re-equip its fleet of aircraft by purchasing three different types:

AIRCRAFT	Cost (millions)	Revenue (millions)	Pilots enabled	For Flights in:	Effort in maintenance Workshop
Embraer 190	33.0	120.0	25	Domestic	1/15
Airbus A350	230.0	300.0	10	Americas and Europe	4/15
Boing 787-8	170.0	260.0	9	Asia	3/15
Available	730.0				≤ 4

- The Boing pilot can pilot all three types of aircraft.
- The Airbus pilot also pilots Embraer.
- The Embraer pilot does not pilot the others.
- In a Boing is required three pilots, the Airbus and Embraer only two.

Required/ Alircraft		Embraer	Airbus	Boing
Pilots	Embraer	2	2	3
	Airbus	0	2	3
	Boing	0	0	3
	Available	25	10	9

The workshops can meet the simultaneous maintenance of 3 Boing 787-8. A Boing requires a maintenance effort equivalent to 3 Embraer and 0.75 Arbus.

It is estimated that 15% of the aircraft are always under preventive maintenance. Formulate the problem of optimization of the purchase of these aircraft by the company.

```
MODEL:
SETS:
PRODUCT:
REVENUE,
PURCHASE,
PILOTS_190,
PILOTS_350,
PILOTS_787,
PILOTS_AVAILABLE,
PRODUCE;
ENDSETS
DATA:
! Products attributes;
PRODUCT =
EMBRAER_190
AIRBUS_A350
BOING_787_8;
! Products attributes          190    350    787;
 REVENUE              =       120    300    260;   ! Expected Revenue in millions;
 PURCHASE             =       33     230    170;   ! Unit purchase cost of aircraft in million;
 PILOTS_190           =       2      2      3;     ! Number of pilots qualified for the Embraer 190;
 PILOTS_350           =       0      2      3;     ! Number of pilots qualified for the Airbus A350;
 PILOTS_787           =       0      0      3;     ! Number of pilots qualified for the Boing 787-8;
 PILOTS_AVAILABLE =           25     10     9;     ! Number of pilots available for these aircraft;
ENDDATA
SUBMODEL MAX6:
[OBJ] MAX = @SUM( PRODUCT(J): Purchase(J) * PRODUCE( J));
! Capital Available for Acquisition;
@SUM(PRODUCT(J): PURCHASE(J) * PRODUCE( J)) <= 730;
! Number of Pilots enable To Pilot or Embraer;
@SUM(PRODUCT(J): PILOTS_190(J) * PRODUCE(j)) <= 25;
! Number of Pilots enable To Pilot or AirBus;
@SUM(PRODUCT(K): PILOTS_350(K) * PRODUCE(K)) <= 10;
! Number of Pilots enable To Pilot or Boing;
@SUM(PRODUCT(L): PILOTS_787(L) * PRODUCE(L)) <= 9;
@FOR(PRODUCT(J): @GIN(PRODUCE(J)));
! Effort in Maintenance Workshops;
1/15*PRODUCE(1) + 4/15*PRODUCE(2) + 3/15*PRODUCE(3) <= 4;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  Expected Revenue in millions:", @NEWLINE( 1));
@TABLE(REVENUE);
@WRITE(" ", @NEWLINE( 1), "  Unit purchase PROFIT of aircraft in million:", @NEWLINE( 1));
@TABLE(PURCHASE);
@WRITE(" ", @NEWLINE( 1), "  Number of pilots qualified for the Embraer 190:", @NEWLINE( 1));
@TABLE(PILOTS_190);
@WRITE(" ", @NEWLINE( 1), "  Number of pilots qualified for the Airbus A350:", @NEWLINE( 1));
@TABLE(PILOTS_350);
@WRITE(" ", @NEWLINE( 1), "  Number of pilots qualified for the Boing 787-8:", @NEWLINE( 1));
@TABLE(PILOTS_787);
@WRITE(" ", @NEWLINE( 1), "   Number of pilots available for these aircraft:", @NEWLINE( 1));
@TABLE(PILOTS_AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX6)
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  PROPOSAL OF PURCHASE  (values in millions) :", @NEWLINE( 2));
@WRITEFOR( PRODUCT(J) : '  [',
        @FORMAT(PRODUCT(J),'-11s'), '] ',
        @FORMAT(PRODUCE(J),'%2.0f'), ' un  x  Purchase: $' ,
        @FORMAT(PURCHASE(J),'%4.0f'), '  = Total: $',
        @FORMAT(PURCHASE(J) * PRODUCE(J),'%3.0f'), @NEWLINE(1), 26*' ','Revenue:  $',
        @FORMAT(REVENUE(J) * PRODUCE(J),'%4.0f'),
@NEWLINE( 2));
@WRITE('  Investment value', 34*' ', '$',@FORMAT(OBJ,'%3.0f'),@NEWLINE(1));
@WRITE('  Gross Revenue', 21*' ', '$', @FORMAT(@SUM(PRODUCT(J):REVENUE(J)*PRODUCE(J)),'%4.0f'), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
Expected Revenue in millions:
 EMBRAER_190  120.0000
 AIRBUS_A350  300.0000
 BOING_787_8  260.0000

Unit purchase PROFIT of aircraft in million:
 EMBRAER_190  33.00000
 AIRBUS_A350  230.0000
 BOING_787_8  170.0000

 Number of pilots qualified for the Embraer 190:
  EMBRAER_190  2.000000
  AIRBUS_A350  2.000000
  BOING_787_8  3.000000

 Number of pilots qualified for the Airbus A350:
  EMBRAER_190  0.000000
  AIRBUS_A350  2.000000
  BOING_787_8  3.000000

 Number of pilots qualified for the Boing 787-8:
  EMBRAER_190  0.000000
  AIRBUS_A350  0.000000
  BOING_787_8  3.000000

  Number of pilots available for these aircraft:
  EMBRAER_190  25.00000
  AIRBUS_A350  10.00000
  BOING_787_8  9.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              730.0000
Objective bound:                              730.0000
Infeasibilities:                              0.000000

 PROPOSAL OF PURCHASE  (values in millions) :

 [EMBRAER_190] 10 un  x  Purchase: $  33  = Total: $330
                         Revenue:  $1200

 [AIRBUS_A350]  1 un  x  Purchase: $ 230  = Total: $230
                         Revenue:  $ 300

 [BOING_787_8]  1 un  x  Purchase: $ 170  = Total: $170
                         Revenue:  $ 260

 Investment value                                 $730
 Gross Revenue                     $1760
```

7

GOAL

Closely related to the newsboy problem (in a mathematical sense) is the airline overbooking problem.

Given that a certain percentage of fliers with reservations will not show up for a flight, airlines that don't overbook will be sending most planes up with empty seats.

Assuming the penalty cost for overbooking is not too high, an airline that hopes to maximize revenue should overbook its flights.

The following model determines the optimal number of reservations to allow on a flight, and assumes the number of no-shows on a flight has a binomial distribution.

This model uses a brut force method to compute the expected profits from overbooking 1 to 6 seats.

Items			Value
1	Total seats available	seat	16
2	Revenue from sold seat	$	225.00
3	Penalty for turned down customer	$	100.00
4	Probability a customer is no-show	%	0.04

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- No Show
- Transport

Source:

- Book 4
- obbok2

```
MODEL:
! A strategy for airlines to minimize the loss from no-shows is to overbook flights.
Too little overbooking results in lost revenue.
Too much overbooking results in excessive penalties.
This model computes expected profits for various levels of overbooking.;
SETS:
   SEAT/1..16/;                    ! seats available ;
   EXTRA/1..6/: EPROFIT;           ! expected profits from overbooking 1-6 seats;
ENDSETS
DATA:
 ! Available data;
   V = 225;                        ! Revenue from a sold seat;
   P = 100;                        ! Penalty for a turned down customer;
   Q = .04;                        ! Probability customer is a no-show;
 ! No. of seats available;
ENDDATA
SUBMODEL S10:
   N = @SIZE( SEAT);
 ! Expected profit with no overbooking;
   EPROFIT0 = V * @SUM( SEAT(I):  (1 - @PBN(1- Q, N, I - 1)));
 ! Expected profit if we overbook by 1 is:
   EPROFIT0 + Prob(he shows) * ( V - (V + P) * Prob(we have no room));
   EPROFIT( 1) = EPROFIT0 + ( 1 - Q) * ( V - ( V + P) * @PBN( Q, N, 0));
 ! In general;
   @FOR( EXTRA( I)| I #GT# 1: EPROFIT( I) = EPROFIT( I - 1) + (1 - Q) * ( V - ( V + P) * @PBN( Q, N + I - 1, I - 1));););
ENDSUBMODEL
CALC:
! Output level (0:verb, 1:terse, 2:only errors, 3:none);
@SET( 'TERSEO',2);
! Post status windows, 1 Yes, 0 No;
@SET( 'STAWIN',0);
! Execute sub-model;
@SOLVE(S10);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Total seats available: ",38*'.', @FORMAT(N,'%5.0f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Revenue from a sold seat: ",35*'.', @FORMAT(V,'%9.2f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Penalty for a turned down customer: ",25*'.', @FORMAT(P,'%6.0f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Probability a customer is a no-show: ",24*'.', @FORMAT(Q,'%7.2f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 2), "  SOLUTION:", @NEWLINE( 1));
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  . This model computes expected profits for various levels of overbooking: ",@NEWLINE( 2));
@WRITE("    Profit with ", 0, ' Overbooking: $',@FORMAT(EPROFIT0,'%7.3f'), @NEWLINE(1));
@FOR(EXTRA(I):
      @WRITE("    Profit with ", I, ' Overbooking: $',@FORMAT(EPROFIT(I),'%7.3f'),
@NEWLINE(1)));
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

. Total seats available: .....................................   16

. Revenue from a sold seat: ..................................   225.00

. Penalty for a turned down customer: ........................   100

. Probability a customer is a no-show: .......................   0.04
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:

. This model computes expected profits for various levels of overbooking:

  Profit with 0 Overbooking: $3456.000
  Profit with 1 Overbooking: $3509.634
  Profit with 2 Overbooking: $3459.355
  Profit with 3 Overbooking: $3373.744
  Profit with 4 Overbooking: $3279.654
  Profit with 5 Overbooking: $3183.953
  Profit with 6 Overbooking: $3087.994
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Operation
- Control
- Ground delays
- Air Traffic

Source:

- Book 4

GOAL

Large airlines tend to base their route structure around the hub concept.

An airline will try to have a large number of flights arrive at the hub airport during a certain short interval time (e.g., 9 a.m. to 10 a.m.) and then have a large number of flights depart the hub shortly thereafter (e.g., 10 a.m. to 11 a.m.).

This allows customers of that airline to travel between a large combination of origin/destination cities with one stop and at most one change of planes.
For example, United Airlines uses one stop and at most one change of planes.

For example, United Airlines uses Chicago as a hub, Delta Airlines uses Atlanta, TWA uses St. Louis, and American uses Dallas/Fort Worth.

A desirable goal in using a hub structure is minimize the amount of changing of planes (and the resulting moving of baggage) at the hub.

This model illustrates how to apply the assignment model:

- A certain airline has six flights arriving at O'Hare airport between 9 and 9:30 a.m.
- The same six airplanes depart on different flights between 9:40 and 10:20 a.m.
- You know the average number of people transferring between incoming and leaving flights.

All the planes are identical. A decision problem is which incoming flight should be assigned to which outgoing flight.

This model helps determine how incoming flights should be assigned to leaving flights so that a minimum number of people need to change planes at the O'Hare stop.

```
MODEL:
SETS:
 FLIGHT;
 ASSIGN( FLIGHT, FLIGHT): X, CHANGE;
ENDSETS
DATA:
 FLIGHT   = 1..6;
! The value of assigning i to j;
 CHANGE =   20     15     16     5     4     7
            17     15     33     12    8     6
            9      12     18     16    30    13
            12     8      11     27    19    14
            -999   7      10     21    10    32
            -999   -999   -999   6     11    13;
ENDDATA
SUBMODEL MAX8:
! Maximize value of assignments;
[OBJ] MAX = @SUM(ASSIGN: X * CHANGE);
@FOR( FLIGHT( I):
        !  Each I must be assigned to some J;
        @SUM( FLIGHT( J): X( I, J)) = 1;
        !  Each I must receive an assignment;
        @SUM( FLIGHT( J): X( J, I)) = 1;  );
ENDSUBMODEL
CALC:
! Output level (0:verb, 1:terse, 2:only errors, 3:none);
@SET( 'TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  Arrival of passengers on six aircraft between 9 and 9:30 am:", @NEWLINE( 1));
@TABLE(CHANGE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Solve the model;
@SOLVE(MAX8);
! Solution report;
@WRITE("  LOGICAL SELECTION: ", @NEWLINE( 1));
@TABLE(X);
@WRITE(" ", @NEWLINE( 2));
@WRITE("  Move pax between aircraft (large combination of origin/destination).", @NEWLINE( 1));
@WRITE("  Passenger departure on six aircraft between 9 and 9:40 am:", @NEWLINE( 2));
@WRITEFOR( ASSIGN(K,L) | X(K,L) #GT# 0: '   PAX: ', CHANGE(K,L),@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX8);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
Arrival of passengers on six aircraft between 9 and 9:30 am:
             1          2          3          4          5          6
 1   20.00000   15.00000   16.00000   5.000000   4.000000   7.000000
 2   17.00000   15.00000   33.00000   12.00000   8.000000   6.000000
 3   9.000000   12.00000   18.00000   16.00000   30.00000   13.00000
 4   12.00000   8.000000   11.00000   27.00000   19.00000   14.00000
 5  -999.0000   7.000000   10.00000   21.00000   10.00000   32.00000
 6  -999.0000  -999.0000  -999.0000   6.000000   11.00000   13.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              135.0000
Infeasibilities:                              0.000000


LOGICAL SELECTION:
            1         2         3         4         5         6
 1  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 2  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000
 3  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000
 4  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000
 5  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 6  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000


Move pax between aircraft (large combination of origin/destination).
Passenger departure on six aircraft between 9 and 9:40 am:

 PAX: 20.00000
 PAX: 33.00000
 PAX: 12.00000
 PAX: 27.00000
 PAX: 32.00000
 PAX: 11.00000
```

9

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Planning
- No Show
- Transport

Source:
- Book 4
- overbook

GOAL

Closely related to the newsboy problem (in a mathematical sense) is the airline overbooking problem.

Given that a certain percentage of fliers with reservations will not show up for a flight, airlines that don't overbook will be sending most planes up with empty seats.

Assuming the penalty cost for overbooking is not too high, an airline that hopes to maximize revenue should overbook its flights.

The following model determines the optimal number of reservations to allow on a flight, and assumes the number of no-shows on a flight has a binomial distribution.

This overbooking model determines the number of reservations, M, to allow on a flight if the no-show distribution is binomial.

Items			Value
1	Total seats available	seat	16
2	Revenue from sold seat	$	225.00
3	Penalty for turned down customer	$	100.00
4	Probability a customer is no-show	%	0.04

```
MODEL:

! This overbooking model determines the number of reservations, M, to allow on a flight if the
 no-show distribution is binomial;
DATA:
! Some available data ;
   N = 16;       ! Total seats available;
   V = 225;      ! Revenue from a sold seat;
   P = 100;      ! Penalty for a turned down customer;
   Q = .04;      ! Probability a customer is a no-show;
ENDDATA
SUBMODEL SM9:
! The probability to turn down customers is @PBN( Q, M, M - N), therefore the corresponding
  expected loss due to imperfect information is:( V + P) * @PBN( Q, M, M - N), and we want the
  loss to equal the revenue V on the margin.  So,
  the break-even equation is:;
   ( V + P) * @PBN( Q, M, M - N) = V;
! Note, you should round up if M is fractional;
ENDSUBMODEL
CALC:
! Output level (0:verb, 1:terse, 2:only errors, 3:none);
@SET( 'TERSEO',2);
! Post status windows, 1 Yes, 0 No;
@SET( 'STAWIN',0);
! Execute sub-model;
@SOLVE(SM9);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Total seats available: ",38*'.', @FORMAT(N,'%5.0f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Revenue from a sold seat: ",35*'.', @FORMAT(V,'%9.2f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Penalty for a turned down customer: ",25*'.', @FORMAT(P,'%6.0f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE("  . Probability a customer is a no-show: ",24*'.', @FORMAT(Q,'%7.2f'), @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 2), "  SOLUTION:", @NEWLINE( 1));
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  . This overbooking model determines the number of reservations: ",
@FORMAT(M,'%7.2f'), @NEWLINE( 2));
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

. Total seats available: ......................................   16

. Revenue from a sold seat: ...................................   225.00

. Penalty for a turned down customer: .........................   100

. Probability a customer is a no-show: ........................   0.04
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:

. This overbooking model determines the number of reservations:   16.52
```

BLOCK 6

Block: TRANSPORT

What should be the route for transporting cargo vehicles so that they deliver the entire cargo in the shortest time and at the lowest total cost?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Consumer
- Logistics

Source:

- Book 2
- Page 231

GOAL

Three warehouse, four Customer Transportation Problem.

Warehouse / Customer		C1	C2	C3	C4	Available
W1	$	6	2	6	7	30 un
W2	$	4	9	5	3	25 un
W3	$	8	8	1	5	21 un
Demand	un	15	17	22	12	-

This model contains report writing statements that ***mimic*** the standard Lingo solution report;

```
MODEL:
CALC:
  ! We need to enable range analysis for this model;
  @SET( 'DUALCO', 2);
ENDCALC
SETS:
  HEADER1 / PROD, LIMIT, VALUE /:;
  WAREHOUSE: CAPACITY;
  CUSTOMER : DEMAND;
  ROUTES( WAREHOUSE, CUSTOMER): COST, VOLUME;
  PXR (WAREHOUSE,HEADER1): SLASUR;
  RXP (CUSTOMER, HEADER1): SLASUR1;
ENDSETS
SUBMODEL TS:
! The objective;
 [OBJ] MIN = @SUM( ROUTES: COST * VOLUME);
! The demand constraints;
 @FOR( CUSTOMER( J): [R_DEM]
        @SUM( WAREHOUSE( I): VOLUME( I, J)) >= DEMAND( J));
! The supply constraints;
 @FOR( WAREHOUSE( I): [R_SUP]
        @SUM( CUSTOMER( J): VOLUME( I, J)) <=  CAPACITY( I));
ENDSUBMODEL
DATA:
! Warehouse attributes;
WAREHOUSE,        CAPACITY =
W1                30
W2                25
W3                21;

! Customer attributes;
CUSTOMER,         DEMAND =
C1                15
C2                17
C3                22
C4                12;

! Cost            C1    C2    C3    C4;
COST =            6,    2,    6,    7,    ! W1;
                  4,    9,    5,    3,    ! W2;
                  8,    8,    1,    5;    ! W3;

INFINITY = 1.E30;
  LENF1S = 18;
  LENF1R = 18;
ENDDATA
```

```
CALC:
! STANDARD REPORT SOLUTION;
 @WRITE( @NEWLINE( 1),'STANDARD REPORT SOLUTION', @NEWLINE( 1));
! Execute sub-model TS;
 @SOLVE(TS);
! MIMIC THE STANDARD LINGO SOLUTION REPORT;
 @WRITE( @NEWLINE( 1),'SIMILAR SOLUTION REPORT', @NEWLINE( 1));
! OBJECTIVE  VALUE;
 @WRITE( 3*' ',
        @IF( @STATUS() #EQ# 0, 'Global optimal', '**** Non-global ***'), ' solution found at iteration:',
             @FORMAT( @ITERS(), '8.14G'),
  @NEWLINE( 1));
  @WRITE( 3*' ', 'Objective value:', @FORMAT( OBJ, '#35.7G'), @NEWLINE( 2));
! HEADER - VARIABLE, VALUE AND REDUCED COST;
  @WRITE( 10*' ', 'Variable          Value       Reduced Cost', @NEWLINE( 1) );
! Variable and Reduced Cost: CAPACITY;
  @WRITEFOR( WAREHOUSE( I):  ( LENF1S - @STRLEN( @NAME( CAPACITY( I))))*' ', @NAME( CAPACITY( I)),
        @FORMAT( CAPACITY( I), '#16.7G'),
        @FORMAT( @DUAL( CAPACITY( I)), '#20.6G'),
  @NEWLINE( 1) );
! Variable and Reduced Cost: DEMAND;
  @WRITEFOR( CUSTOMER( I):
        ( LENF1S - @STRLEN( @NAME( DEMAND( I))))*' ', @NAME( DEMAND( I)),
        @FORMAT( DEMAND( I), '#16.7G'),
        @FORMAT( @DUAL( DEMAND( I)), '#20.6G'),
  @NEWLINE( 1));
! Variable and Reduced Cost: COST;
   @WRITEFOR( ROUTES( I, J):
        ( LENF1S - @STRLEN( @NAME( COST( I, J))))*' ', @NAME( COST( I, J)),
        @FORMAT( COST( I, J), '#16.7G'),
        @FORMAT( @DUAL( COST( I, J)), '#20.6G'),
   @NEWLINE( 1));
! Variable and Reduced Cost: VOLUME;
   @WRITEFOR( ROUTES( I, J): ( LENF1S - @STRLEN( @NAME( VOLUME( I, J))))*' ', @NAME( VOLUME( I, J)),
        @FORMAT( VOLUME( I, J), '#16.7G'),
        @FORMAT( @DUAL( VOLUME( I, J)), '#20.7G'),
   @NEWLINE( 1));
! HEADER - SLACK OR SURPLUS AND DUAL PRICE;
  @WRITE( @NEWLINE( 1), 15*' ','Row    Slack or Surplus      Dual Price', @NEWLINE( 1) );
  @WRITE(15*' ','OBJ',
        @FORMAT( OBJ, '#16.7G'),
        @FORMAT( @DUAL( OBJ), '#20.7G'),
  @NEWLINE( 1));
! Slack or Surplus and Dual Price: DEMAND;
   @WRITEFOR( CUSTOMER( I): ( LENF1S - @STRLEN( @NAME( R_DEM( I))))*' ', @NAME( R_DEM( I)),
         @FORMAT( R_DEM( I), '#16.7G'),
        @FORMAT( @DUAL( R_DEM( I)), '#20.7G'),
   @NEWLINE( 1));
! Slack or Surplus and Dual Price: CAPACITY;
    @WRITEFOR( WAREHOUSE( I): ( LENF1S - @STRLEN( @NAME( R_SUP( I))))*' ', @NAME( R_SUP( I)),
        @FORMAT( R_SUP( I), '#16.7G'),
        @FORMAT( @DUAL( R_SUP( I)), '#20.7G'),
   @NEWLINE( 1));
```

```
! MIMIC THE STANDARD LINGO RANGE REPORT;
! Header;
  @WRITE( @NEWLINE( 1),  ' Ranges in which the basis is unchanged:', @NEWLINE( 2));
! Objective Coefficient Ranges;
  @WRITE( ' ', 'Objective Coefficient Ranges', @NEWLINE( 1));
  @WRITE( 28*' ', 'Current       Allowable       Allowable', @NEWLINE( 1));
  @WRITE( 10*' ', 'Variable     Coefficient       Increase        Decrease', @NEWLINE( 1));
! COST/VOLUME;
 ! @TEXT() = ;
    @WRITEFOR( ROUTES( I, J): ( LENF1R - @STRLEN( @NAME( VOLUME( I, J))))*' ',
        @NAME( VOLUME( I, J)), @FORMAT( COST( I, J), '#17.7G'),
        @IF( @RANGEU( VOLUME( I, J)) #LT# INFINITY,
                @FORMAT( @RANGEU( VOLUME( I, J)), '#17.7G'),'        INFINITY'),
                @IF( @RANGED( VOLUME( I, J)) #LT# INFINITY,
                        @FORMAT( @RANGED( VOLUME( I, J)), '#17.7G'),'        INFINITY'),
   @NEWLINE( 1) );
! Righthand Side Ranges:
  @WRITE( @NEWLINE( 1), ' ', 'Righthand Side Ranges', @NEWLINE( 1));
  @WRITE( 15*' ','Row         Current       Allowable       Allowable', @NEWLINE( 1));
  @WRITE( 32*' ', 'RHS         Increase        Decrease', @NEWLINE( 1));
! DEMAND;
   @WRITEFOR( CUSTOMER( I): ( LENF1R - @STRLEN( @NAME( R_DEM( I))))*' ',
        @NAME( R_DEM( I)), @FORMAT( DEMAND( I), '#17.7G'),
        @IF( @RANGEU( R_DEM( I)) #LT# INFINITY,
                @FORMAT( @RANGEU( R_DEM( I)), '#17.7G'), '        INFINITY'),
                @FORMAT( @RANGED( R_DEM( I)), '#17.7G'),
   @NEWLINE( 1));
! CAPACITY;
   @WRITEFOR( WAREHOUSE( I): ( LENF1R - @STRLEN( @NAME( R_SUP( I))))*' ',  @NAME( R_SUP( I)),
         @FORMAT( CAPACITY( I), '#17.7G'),
         @IF( @RANGEU( R_SUP( I)) #LT# INFINITY,
                @FORMAT( @RANGEU( R_SUP( I)), '#17.7G'), '        INFINITY'),
                @FORMAT( @RANGED( R_SUP( I)), '#17.7G'),
    @NEWLINE( 1));
```

```
! COMPACT REPORT SOLUTION;
@WRITE( @NEWLINE( 1),'COMPACT REPORT SOLUTION', @NEWLINE( 1));
! Output level: 0=Verbose, 1-Terse;
 @SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
 @SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE:", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND:", @NEWLINE( 1));
@TABLE(DEMAND);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From: ',
     @FORMAT(WAREHOUSE( I),'-3s'), '  Ship to: ' ,
     @FORMAT(CUSTOMER( J),'-3s'), '  ',
     @FORMAT(VOLUME( I, J),'%2.0f'),'Un  x  Unit cost: $',
     @FORMAT(COST( I, J),'%3.2f') ,' = Total: $',
     @FORMAT(COST( I, J) * VOLUME(I,J),'%6.2f'),
@NEWLINE( 1));
! Slack/Surplus Product report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = CONSUMER DEMAND (un): ", @NEWLINE( 1));
@FOR(RXP(L,C):
        SLASUR1(L,2) = DEMAND(L);
        SLASUR1(L,1) = SLASUR1(L,2) -  R_DEM(L);
        SLASUR1(L,3) = SLASUR1(L,1) - SLASUR1(L,2););
@TABLE(SLASUR1);
! Slack/Surplus Product report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = WAREHOUSES CAPACITY (un): ", @NEWLINE( 1));
@FOR(PXR(L,C):
        SLASUR(L,2) = CAPACITY(L);
        SLASUR(L,1) = SLASUR(L,2) - R_SUP(L);
        SLASUR(L,3) = SLASUR(L,1) - SLASUR(L,2););
@TABLE(SLASUR);
@WRITE( @NEWLINE( 1),'  ALTERNATIVE TO GENERATE THE SCALAR MODEL', @NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(TS);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
       Variable           Value        Reduced Cost
       INFINITY       0.1000000E+31        0.000000
         LENF1S        18.00000            0.000000
         LENF1R        18.00000            0.000000
  CAPACITY( W1)        30.00000            0.000000
  CAPACITY( W2)        25.00000            0.000000
  CAPACITY( W3)        21.00000            0.000000
    DEMAND( C1)        15.00000            0.000000
    DEMAND( C2)        17.00000            0.000000
    DEMAND( C3)        22.00000            0.000000
    DEMAND( C4)        12.00000            0.000000
  COST( W1, C1)        6.000000            0.000000
  COST( W1, C2)        2.000000            0.000000
  COST( W1, C3)        6.000000            0.000000
  COST( W1, C4)        7.000000            0.000000
  COST( W2, C1)        4.000000            0.000000
  COST( W2, C2)        9.000000            0.000000
  COST( W2, C3)        5.000000            0.000000
  COST( W2, C4)        3.000000            0.000000
  COST( W3, C1)        8.000000            0.000000
  COST( W3, C2)        8.000000            0.000000
  COST( W3, C3)        1.000000            0.000000
  COST( W3, C4)        5.000000            0.000000
VOLUME( W1, C1)        2.000000            0.000000
VOLUME( W1, C2)        17.00000            0.000000
VOLUME( W1, C3)        1.000000            0.000000
VOLUME( W1, C4)        0.000000            2.000000
VOLUME( W2, C1)        13.00000            0.000000
VOLUME( W2, C2)        0.000000            9.000000
VOLUME( W2, C3)        0.000000            1.000000
VOLUME( W2, C4)        12.00000            0.000000
VOLUME( W3, C1)        0.000000            7.000000
VOLUME( W3, C2)        0.000000            11.00000
VOLUME( W3, C3)        21.00000            0.000000
VOLUME( W3, C4)        0.000000            5.000000
```

❖ SOLUTION - STANDARD REPORT SOLUTION

```
STANDARD REPORT SOLUTION
  Global optimal solution found.
  Objective value:                              161.0000
  Infeasibilities:                              0.000000
  Total solver iterations:                             6
  Elapsed runtime seconds:                          0.05

  Model Class:                                        LP

  Total variables:                                    12
  Nonlinear variables:                                 0
  Integer variables:                                   0
  Total constraints:                                   8
  Nonlinear constraints:                               0
  Total nonzeros:                                     36
  Nonlinear nonzeros:                                  0


                  Row    Slack or Surplus      Dual Price
                  OBJ        161.0000           -1.000000
             R_DEM( C1)      0.000000           -6.000000
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
          Variable           Value        Reduced Cost
     CAPACITY( W1)        30.00000             0.00000
     CAPACITY( W2)        25.00000             0.00000
     CAPACITY( W3)        21.00000             0.00000
       DEMAND( C1)        15.00000             0.00000
       DEMAND( C2)        17.00000             0.00000
       DEMAND( C3)        22.00000             0.00000
       DEMAND( C4)        12.00000             0.00000
     COST( W1, C1)        6.000000             0.00000
     COST( W1, C2)        2.000000             0.00000
     COST( W1, C3)        6.000000             0.00000
     COST( W1, C4)        7.000000             0.00000
     COST( W2, C1)        4.000000             0.00000
     COST( W2, C2)        9.000000             0.00000
     COST( W2, C3)        5.000000             0.00000
     COST( W2, C4)        3.000000             0.00000
     COST( W3, C1)        8.000000             0.00000
     COST( W3, C2)        8.000000             0.00000
     COST( W3, C3)        1.000000             0.00000
     COST( W3, C4)        5.000000             0.00000
   VOLUME( W1, C1)        2.000000            0.000000
   VOLUME( W1, C2)        17.00000            0.000000
   VOLUME( W1, C3)        1.000000            0.000000
   VOLUME( W1, C4)        0.000000            2.000000
   VOLUME( W2, C1)        13.00000            0.000000
   VOLUME( W2, C2)        0.000000            9.000000
   VOLUME( W2, C3)        0.000000            1.000000
   VOLUME( W2, C4)        12.00000            0.000000
   VOLUME( W3, C1)        0.000000            7.000000
   VOLUME( W3, C2)        0.000000            11.00000
   VOLUME( W3, C3)        21.00000            0.000000
   VOLUME( W3, C4)        0.000000            5.000000
```

❖ SOLUTION - SIMILAR SOLUTION REPORT

```
SIMILAR SOLUTION REPORT
   Global optimal solution found at iteration:       6
   Objective value:                           161.0000

                Row    Slack or Surplus      Dual Price
                OBJ          161.0000       -1.000000
        R_DEM( C1)         -0.000000       -6.000000
        R_DEM( C2)         -0.000000       -2.000000
        R_DEM( C3)         -0.000000       -6.000000
        R_DEM( C4)         -0.000000       -5.000000
        R_SUP( W1)          10.00000       -0.000000
        R_SUP( W2)          0.000000        2.000000
        R_SUP( W3)          0.000000        5.000000
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

Variable	Value	Reduced Cost
CAPACITY(W1)	30.00000	0.00000
CAPACITY(W2)	25.00000	0.00000
CAPACITY(W3)	21.00000	0.00000
DEMAND(C1)	15.00000	0.00000
DEMAND(C2)	17.00000	0.00000
DEMAND(C3)	22.00000	0.00000
DEMAND(C4)	12.00000	0.00000
COST(W1, C1)	6.000000	0.00000
COST(W1, C2)	2.000000	0.00000
COST(W1, C3)	6.000000	0.00000
COST(W1, C4)	7.000000	0.00000
COST(W2, C1)	4.000000	0.00000
COST(W2, C2)	9.000000	0.00000
COST(W2, C3)	5.000000	0.00000
COST(W2, C4)	3.000000	0.00000
COST(W3, C1)	8.000000	0.00000
COST(W3, C2)	8.000000	0.00000
COST(W3, C3)	1.000000	0.00000
COST(W3, C4)	5.000000	0.00000

❖ SOLUTION - SIMILAR RANGE REPORT

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

Variable	Current Coefficient	Allowable Increase	Allowable Decrease
VOLUME(W1, C1)	6.000000	0.000000	0.000000
VOLUME(W1, C2)	2.000000	0.000000	0.000000
VOLUME(W1, C3)	6.000000	0.000000	0.000000
VOLUME(W1, C4)	7.000000	0.000000	0.000000
VOLUME(W2, C1)	4.000000	0.000000	0.000000
VOLUME(W2, C2)	9.000000	0.000000	0.000000
VOLUME(W2, C3)	5.000000	0.000000	0.000000
VOLUME(W2, C4)	3.000000	0.000000	0.000000
VOLUME(W3, C1)	8.000000	0.000000	0.000000
VOLUME(W3, C2)	8.000000	0.000000	0.000000
VOLUME(W3, C3)	1.000000	0.000000	0.000000
VOLUME(W3, C4)	5.000000	0.000000	0.000000

Righthand Side Ranges

Row	Current RHS	Allowable Increase	Allowable Decrease
R_DEM(C1)	15.00000	0.000000	0.000000
R_DEM(C2)	17.00000	0.000000	0.000000
R_DEM(C3)	22.00000	0.000000	0.000000
R_DEM(C4)	12.00000	0.000000	0.000000
R_SUP(W1)	30.00000	0.000000	0.000000
R_SUP(W2)	25.00000	0.000000	0.000000
R_SUP(W3)	21.00000	0.000000	0.000000

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
COMPACT REPORT SOLUTION
  DATA:
  SHIPPING COST:
              C1        C2        C3        C4
   W1  6.000000  2.000000  6.000000  7.000000
   W2  4.000000  9.000000  5.000000  3.000000
   W3  8.000000  8.000000  1.000000  5.000000

  AVAILABLE:
   W1  30.00000
   W2  25.00000
   W3  21.00000

  DEMAND:
   C1  15.00000
   C2  17.00000
   C3  22.00000
   C4  12.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
IDEAL TRANSPORT PROGRAM:
From: W1   Ship to: C1    2Un  x  Unit cost: $6.00 = Total: $ 12.00
From: W1   Ship to: C2   17Un  x  Unit cost: $2.00 = Total: $ 34.00
From: W1   Ship to: C3    1Un  x  Unit cost: $6.00 = Total: $  6.00
From: W2   Ship to: C1   13Un  x  Unit cost: $4.00 = Total: $ 52.00
From: W2   Ship to: C4   12Un  x  Unit cost: $3.00 = Total: $ 36.00
From: W3   Ship to: C3   21Un  x  Unit cost: $1.00 = Total: $ 21.00

SLACK/SURPLUS LIMIT = CONSUMER DEMAND (un):
         PROD      LIMIT      VALUE
 C1  15.00000  15.00000  0.000000
 C2  17.00000  17.00000  0.000000
 C3  22.00000  22.00000  0.000000
 C4  12.00000  12.00000  0.000000

SLACK/SURPLUS LIMIT = WAREHOUSES CAPACITY (un):
          PROD       LIMIT       VALUE
 W1   20.00000   30.00000   -10.00000
 W2   25.00000   25.00000    0.000000
 W3   21.00000   21.00000    0.000000
```

2

GOAL

A mining company wants to minimize the use of trucks carrying ore (from where it is removed) and deposits (where it is stored).

The table below gives the distances involved in feet:

Destination		Deposits		Available	
Origin		D1	D2	Minimum (ton)	Maximum (ton)
A	feet	300	400	20,000	40,000
B	feet	600	700	40,000	60,000
C	feet	800	300	45,000	60,000
Capacity	Ton	50,000	60,000	-	-

The maximum capacity of receiving deposits 1 and 2 are respectively 50,000 and 60,000 tons. Knowing that each truck journey carries 100 tons, we ask for the transportation scheme that minimizes the total distance covered

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Delivery
- Distribution
- Consumer
- Logistics

Source:
- Book 1
- Page 81

```
MODEL:
SETS:
ORIGIN: AVAILABLE_MIN, AVAILABLE_MAX;
DESTINATION: CAPACITY ;
ROUTES( ORIGIN, DESTINATION): DISTANCE, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
ORIGIN,          AVAILABLE_MIN,    AVAILABLE_MAX =
A                20000             40000
B                40000             60000
C                45000             60000;
! Consumer attributes;
DESTINATION,     CAPACITY  =
D1               50000
D2               60000;
! Distance       D1     D2;
DISTANCE   =     300    400    ! A;
                 600    700    ! B;
                 800    300;   ! C;
ENDDATA
SUBMODEL  MIN2:
[OBJ] MIN = @SUM( ROUTES( I, J): DISTANCE( I, J) * VOLUME( I, J));
! Origin A constraints;
@SUM(DESTINATION(J):VOLUME(1,J)) >= AVAILABLE_MIN(1)/100;
@SUM(DESTINATION(J):VOLUME(1,1)) <= AVAILABLE_MAX(1)/100;
! Origin B constraints;
@SUM(DESTINATION(J): VOLUME(2,J)) >=  AVAILABLE_MIN(2)/100;
@SUM(DESTINATION(J): VOLUME(2,J)) <=  AVAILABLE_MIN(2)/100;
! Origin C constraints;
@SUM(DESTINATION(J): VOLUME(3,J)) >=  AVAILABLE_MIN(3)/100;
@SUM(DESTINATION(J): VOLUME(3,J)) <=  AVAILABLE_MIN(3)/100;
! The Capacity constraints;
@SUM(ORIGIN(I): VOLUME(I,1)) <= CAPACITY(1)/100;
@SUM(ORIGIN(I): VOLUME(I,2)) <= CAPACITY(2)/100;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
 @SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
 @SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  DISTANCE (feet):", @NEWLINE( 1));
@TABLE(DISTANCE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE MINIMUM (Ton):", @NEWLINE( 1));
@TABLE(AVAILABLE_MIN);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE MAXIMUM (Ton):", @NEWLINE( 1));
@TABLE(AVAILABLE_MAX);
@WRITE(" ", @NEWLINE( 1), "  CAPACITY (Ton):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( K, L) | VOLUME(K,L) #GT# 0: '  ',@NAME(VOLUME(K,L)),
        @FORMAT(VOLUME( K, L),'%4.0f'), ' Travel x Distance:',
        @FORMAT(DISTANCE(K, L),'%4.0f'),'Feet = ',
        @FORMAT(DISTANCE (K, L) * VOLUME(K,L),'%6.0f'),
@NEWLINE( 1));
@WRITE( "  Total:", 8*' ',
        @SUM(ROUTES(K,L): VOLUME(K,L))," Travel", 23*' ',
        @SUM(ROUTES(K,L): DISTANCE (K, L) * VOLUME(K,L)), @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
DISTANCE (feet):
          D1        D2
 A  300.0000  400.0000
 B  600.0000  700.0000
 C  800.0000  300.0000

AVAILABLE MINIMUM (Ton):
 A  20000.00
 B  40000.00
 C  45000.00

AVAILABLE MAXIMUM (Ton):
 A  40000.00
 B  60000.00
 C  60000.00

CAPACITY (Ton):
 D1  50000.00
 D2  60000.00
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              445000.0
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
VOLUME( A, D1) 100 Travel x Distance: 300Feet =  30000
VOLUME( A, D2) 100 Travel x Distance: 400Feet =  40000
VOLUME( B, D1) 400 Travel x Distance: 600Feet = 240000
VOLUME( C, D2) 450 Travel x Distance: 300Feet = 135000
Total:        1050 Travel                       445000
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Distribution
- Consumer
- Logistics

GOAL

A company has two branches and provides the delivery service to six different store.

Answer which customers to attend, from each affiliate, in order to minimize their cost of delivery.

All the data for the development of the model are described below:

Shipping		Store 1	Store 2	Store 3	Store 4	Store 5	Store 6	Available (un)
Branch 1	$	7.00	9.00	1.00	12.00	7.00	4.00	2,500
Branch 2	$	4.00	5.00	12.00	1.00	3.00	8.00	2,000
Demand	un	1400	1560	300	150	570	520	-

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): COST, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
SUPPLIER,          AVAILABLE   =
BRANCH1            2500
BRANCH2            2000;
! Consumer attributes;
CONSUMER,          DEMAND      =
STORE1             1400
STORE2             1560
STORE3             300
STORE4             150
STORE5             570
STORE6             520;
! Cost             STO1  STO2  STO3  STO4  STO5  STO6;
COST   =           7     9     1     12    7     4          ! Branch1;
                   4     5     12    1     3     8;         ! Branch2;
ENDDATA
SUBMODEL  MIN3:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
      [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The available constraints;
@FOR( SUPPLIER( I):
      [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
 @SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
 @SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), " SHIPPING COST ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Unity):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (Unity):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ', SUPPLIER( I), ' Ship ',
      @FORMAT(VOLUME( I, J),'%4.0f'), 'Un  To ',
      @FORMAT(CONSUMER( J),'-6s'), '  Shipping cost: $',
      @FORMAT(COST (I, J),'%4.2f'), ' Total: $',
      @FORMAT(COST (I, J) * VOLUME(I,J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
 DATA:
SHIPPING COST ($):
             STORE1    STORE2    STORE3    STORE4    STORE5    STORE6
  BRANCH1  7.000000  9.000000  1.000000  12.00000  7.000000  4.000000
  BRANCH2  4.000000  5.000000  12.00000  1.000000  3.000000  8.000000

 AVAILABLE (Unity):
  BRANCH1  2500.000
  BRANCH2  2000.000

 DEMAND (Unity):
  STORE1  1400.000
  STORE2  1560.000
  STORE3  300.0000
  STORE4  150.0000
  STORE5  570.0000
  STORE6  520.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                            22960.00
Infeasibilities:                                            0.000000

IDEAL TRANSPORT PROGRAM:
BRANCH1 Ship 1400Un  To STORE1  Shipping cost: $7.00 Total: $9800.00
BRANCH1 Ship  300Un  To STORE3  Shipping cost: $1.00 Total: $ 300.00
BRANCH1 Ship  280Un  To STORE5  Shipping cost: $7.00 Total: $1960.00
BRANCH1 Ship  520Un  To STORE6  Shipping cost: $4.00 Total: $2080.00
BRANCH2 Ship 1560Un  To STORE2  Shipping cost: $5.00 Total: $7800.00
BRANCH2 Ship  150Un  To STORE4  Shipping cost: $1.00 Total: $ 150.00
BRANCH2 Ship  290Un  To STORE5  Shipping cost: $3.00 Total: $ 870.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Consumer
- Logistics

GOAL

Three warehouses supply five stores, each warehouse is limited to the total number of units to be shipped and the demand of the Stores too.

All the data for the development of the model are described below:

Shipping			Store 1	Store 2	Store 3	Store 4	Store 5	Available (un)
Warehouse	1	$	16.00	14.00	12.00	12.00	16.00	170
	2	$	12.00	4.00	14.00	8.00	8.00	60
	3	$	8.00	6.00	4.00	14.00	10.00	90
Demand		un	23	69	76	70	82	-

Determine the amount that should be sent from each deposit to each Store in order to minimize the cost of shipping.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): COST, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
SUPPLIER,        AVAILABLE  =
WAREHOUSE1       170
WAREHOUSE2       60
WAREHOUSE3       90;
! Consumer attributes;
CONSUMER,        DEMAND   =
STORE1           23
STORE2           69
STORE3           76
STORE4           70
STORE5           82;
! Cost values    STO1  STO2  STO3  STO4  STO5;
COST   =         16    14    12    12    16          ! WAREHOUSE1;
                 12    4     14    8     8           ! WAREHOUSE2;
                  8    6     4     14    10;         ! WAREHOUSE3;
ENDDATA
SUBMODEL MIN4:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
      [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The capacity constraints;
@FOR( SUPPLIER( I):
       [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
 @SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
 @SET('STAWIN',0);
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING COST ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (un):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN4);
! Solution report;
@WRITE("  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1) );
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
        @FORMAT(SUPPLIER( I), '-10s'),' Ship ',
        @FORMAT(VOLUME( I, J),'%3.0f'),'Un  To ',
        @FORMAT(CONSUMER( J), '-7s'),'x Shipping cost: $',
        @FORMAT(COST (I, J),'%5.2f'),' = Total: $',
        @FORMAT(VOLUME(I,J) * COST(I,J), '%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN4;
ENDCALC
END
```

DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING COST ($):
               STORE1      STORE2      STORE3       STORE4      STORE5
 WAREHOUSE1  16.00000    14.00000    12.00000     12.00000    16.00000
 WAREHOUSE2  12.00000    4.000000    14.00000     8.000000    8.000000
 WAREHOUSE3  8.000000    6.000000    4.000000     14.00000    10.00000

AVAILABLE (Unity):
 WAREHOUSE1  170.0000
 WAREHOUSE2  60.00000
 WAREHOUSE3  90.00000

DEMAND (Un):
 STORE1      23.00000
 STORE2      69.00000
 STORE3      76.00000
 STORE4      70.00000
 STORE5      82.00000
```

SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                  3078.000
Infeasibilities:                                                  0.000000

IDEAL TRANSPORT PROGRAM:
WAREHOUSE1 Ship  18Un  To STORE3 x Shipping cost: $12.00 = Total: $ 216.00
WAREHOUSE1 Ship  70Un  To STORE4 x Shipping cost: $12.00 = Total: $ 840.00
WAREHOUSE1 Ship  82Un  To STORE5 x Shipping cost: $16.00 = Total: $1312.00
WAREHOUSE2 Ship  60Un  To STORE2 x Shipping cost: $ 4.00 = Total: $ 240.00
WAREHOUSE3 Ship  23Un  To STORE1 x Shipping cost: $ 8.00 = Total: $ 184.00
WAREHOUSE3 Ship   9Un  To STORE2 x Shipping cost: $ 6.00 = Total: $  54.00
WAREHOUSE3 Ship  58Un  To STORE3 x Shipping cost: $ 4.00 = Total: $ 232.00
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Consumer
- Logistics

GOAL

A company must schedule shipment schedules for its products, which are shipped from three factories to four warehouses located at strategic points on the market.

Taking into account the type of transport that can be used in each case, as well as the distances between the factories and the warehouses, the costs are differentiated for each factory / warehouse combination.

All the data for the development of the model are described below:

Destiny			W1	W2	W3	W4	Available (un)
Plant	1	$	8.00	14.00	14.00	2.00	200
	2	$	24.00	16.00	16.00	16.00	400
	3	$	16.00	32.00	32.00	10.00	300
Demand		un	160	180	240	320	-

Determine the amount that should be shipped from each factory to each warehouse in order to minimize the cost of shipping.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): COST, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
SUPPLIER,           AVAILABLE       =
PLANT1              200
PLANT2              400
PLANT3              300;
! Consumer attributes;
CONSUMER,           DEMAND    =
WAREHOUSE1          160
WAREHOUSE2          180
WAREHOUSE3          240
WAREHOUSE4          320;

! Cost attributes   WRH1 WRH2 WRH3 WRH4;
COST    =           8    14   14   2          ! PLANT1;
                    24   16   16   16         ! PLANT2;
                    16   32   32   10;        ! PLANT3;
ENDDATA
SUBMODEL MIN5:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
      [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The capacity constraints;
@FOR( SUPPLIER( I):
      [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
 @SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
 @SET('STAWIN',0);
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING COST ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (ton):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (ton):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN5);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
      @FORMAT(SUPPLIER( I),'-6s'), ' Ship ',
      @FORMAT(VOLUME( I, J),'%3.0f'), 'ton To ',
      @FORMAT(CONSUMER( J),'-10s'), '  Shipping cost: $',
      @FORMAT(COST (I, J) * VOLUME(I,J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING COST ($):
           WAREHOUSE1  WAREHOUSE2  WAREHOUSE3  WAREHOUSE4
 PLANT1      8.000000    14.00000    14.00000    2.000000
 PLANT2      24.00000    16.00000    16.00000    16.00000
 PLANT3      16.00000    32.00000    32.00000    10.00000

AVAILABLE (ton):
 PLANT1      200.0000
 PLANT2      400.0000
 PLANT3      300.0000

DEMAND (ton):
 WAREHOUSE1  160.0000
 WAREHOUSE2  180.0000
 WAREHOUSE3  240.0000
 WAREHOUSE4  320.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                  11000.00
Infeasibilities:                                  0.000000


 IDEAL TRANSPORT PROGRAM:
PLANT1 Ship  20ton To WAREHOUSE2  Shipping cost: $ 280.00
PLANT1 Ship 180ton To WAREHOUSE4  Shipping cost: $ 360.00
PLANT2 Ship 160ton To WAREHOUSE2  Shipping cost: $2560.00
PLANT2 Ship 240ton To WAREHOUSE3  Shipping cost: $3840.00
PLANT3 Ship 160ton To WAREHOUSE1  Shipping cost: $2560.00
PLANT3 Ship 140ton To WAREHOUSE4  Shipping cost: $1400.00
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Consumer
- Logistics

Source:

- Book 2
- Page 192

GOAL

A construction company has signed a contract to build 4 lots in different cities of the state. Each building requires a large amount of cement to be sent to each building.

The developer can buy from more than one company for the same project. All the data for the development of the model are described below:

Building			B1	B2	B3	B4	Available (ton)
Provider	1	$	120.00	115.00	130.00	125.00	525
	2	$	100.00	150.00	110.00	105.00	450
	3	$	40.00	95.00	145.00	165.00	550
Demand		ton	450	275	300	350	-

The problem is to determine how much to buy from each supplier in order to meet the demands at the lowest cost possible.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): COST, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
SUPPLIER ,        AVAILABLE        =
PROVIDER1         525
PROVIDER2         450
PROVIDER3         550;
! Consumer attributes;
CONSUMER ,        DEMAND     =
BUILDING1         450
BUILDING2         275
BUILDING3         300
BUILDING4         350;
! Cost value      B1    B2    B3    B3;
COST  =           120   115   130   125   ! PROVIDER1;
                  100   150   110   105   ! PROVIDER2;
                  40     95   145   165;  ! PROVIDER3;
ENDDATA
SUBMODEL MIN6:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
       [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The Available constraints;
@FOR( SUPPLIER( I):
        [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING COST ($):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (ton):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (ton):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN6);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
        @FORMAT(SUPPLIER( I), '-10s'),' Ship ',
        @FORMAT(VOLUME( I, J),'%2.0f'), 'ton To ',
        @FORMAT(CONSUMER( J), '-10s'),' x Shipping cost $',
        @FORMAT(COST (I, J),'%6.2f'), ' = Total: $',
        @FORMAT(COST (I, J) * VOLUME(I,J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING COST ($):
            BUILDING1  BUILDING2  BUILDING3  BUILDING4
 PROVIDER1   120.0000   115.0000   130.0000   125.0000
 PROVIDER2   100.0000   150.0000   110.0000   105.0000
 PROVIDER3   40.00000   95.00000   145.0000   165.0000

AVAILABLE (ton):
 PROVIDER1  525.0000
 PROVIDER2  450.0000
 PROVIDER3  550.0000

DEMAND (ton):
 BUILDING1  450.0000
 BUILDING2  275.0000
 BUILDING3  300.0000
 BUILDING4  350.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                    121375.0
Infeasibilities:                                                    0.000000

IDEAL TRANSPORT PROGRAM:
PROVIDER1  Ship 175ton To BUILDING2  x Shipping cost $115.00 = Total: $20125.00
PROVIDER1  Ship 200ton To BUILDING3  x Shipping cost $130.00 = Total: $26000.00
PROVIDER2  Ship 100ton To BUILDING3  x Shipping cost $110.00 = Total: $11000.00
PROVIDER2  Ship 350ton To BUILDING4  x Shipping cost $105.00 = Total: $36750.00
PROVIDER3  Ship 450ton To BUILDING1  x Shipping cost $ 40.00 = Total: $18000.00
PROVIDER3  Ship 100ton To BUILDING2  x Shipping cost $ 95.00 = Total: $ 9500.00
```

7

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Supplier
- Logistics

Source:

- Book 2
- Page 166

GOAL

A producer and distributor of citrus products that has three orange producing farms in Farm 1, Farm 2 and Farm 3, 200,000, 600,000 and 225,000 boxes of oranges are available in these locations respectively.

The processing plants for oranges are in City A, City B and City C, where the demand of each plant is 275,000, 400,000 and 300,000 respectively of boxes. All the data for the development of the model are described below:

Farmer / City		A	B	C	Available (box)
F1	km	120	115	130	200,000
F2	km	100	150	110	450,000
F3	km	40	95	145	550,000
Demand	box	275,000	400,000	300,000	-

The company hires a local carrier that takes the orange boxes from the farms to the processing units.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): ROUTE, VOLUME;
ENDSETS
DATA:
!Supplier attributes;
SUPPLIER,           AVAILABLE    =
FARM_1              200000
FARM_2              600000
FARM_3              225000;
! Consumer attributes;
CONSUMER,           DEMAND       =
CITY_A              275000
CITY_B              400000
CITY_C              300000;
! Route             CITY_A       CITY_B       CITY_C;
ROUTE    =          21           50           40           ! FARM_1;
                    35           30           22           ! FARM_2;
                    55           20           25;          ! FARM_3;
ENDDATA
SUBMODEL MIN7:
MIN = @SUM( ROUTES( I, J): ROUTE( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( SUPPLIER( J):
        [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The capacity constraints;
@FOR( CONSUMER( I):
        [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING BOXES:", @NEWLINE( 1));
@TABLE(ROUTE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Boxes): ", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (Boxes):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN7);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  ',
        @FORMAT(SUPPLIER( I),'-6s'), '  Shipping to  ',
        @FORMAT(CONSUMER( J),'6s'),' ',
        @FORMAT(VOLUME( I, J),'%8.0f'), ' Box x Distance: ',
        @FORMAT(ROUTE(I,J),'%2.0f'),' Km  = ',
        @FORMAT(VOLUME(i,J) * ROUTE(I, J),'%8.0f'),' Km',
@NEWLINE( 1));
@WRITE('  TOTAL DISTANCE:',13*' ',
        @FORMAT(@SUM(ROUTES(I,J):VOLUME(i,J)),'%8.0f'),' Box', 22*' ',
        @FORMAT(@SUM(ROUTES(I,J):VOLUME(i,J) * ROUTE(I, J)),'%8.0f'),' Km', @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
DISTANCE  ( Km ):
           CITY_A    CITY_B    CITY_C
 FARM_1  21.00000  50.00000  40.00000
 FARM_2  35.00000  30.00000  22.00000
 FARM_3  55.00000  20.00000  25.00000

AVAILABLE (Boxes):
 FARM_1  200000.0
 FARM_2  600000.0
 FARM_3  225000.0

DEMAND (Boxes):
 CITY_A  275000.0
 CITY_B  400000.0
 CITY_C  300000.0
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                              23175000
Infeasibilities:                                              0.000000

IDEAL TRANSPORT PROGRAM:
FARM_1  Shipping to  CITY_A   200000 Box x Distance: 21 Km  =  4200000 Km
FARM_2  Shipping to  CITY_A    75000 Box x Distance: 35 Km  =  2625000 Km
FARM_2  Shipping to  CITY_B   175000 Box x Distance: 30 Km  =  5250000 Km
FARM_2  Shipping to  CITY_C   300000 Box x Distance: 22 Km  =  6600000 Km
FARM_3  Shipping to  CITY_B   225000 Box x Distance: 20 Km  =  4500000 Km
TOTAL DISTANCE:               975000 Box                      23175000 Km
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Distribution
- Logistics

GOAL

Three factories 1,2 and 3 wish to transport their products to four different A, B, C, and D Warehouses. Information on costs, manufacturing capacity and demand can be found bellow:

Cost	To		WA	WB	WC	WD	Capacity (un)
From	P1	$	2.00	5.00	4.00	8.00	400
	P2	$	3.00	2.00	5.00	4.00	500
	P3	$	5.00	3.00	9.00	5.00	700
Demand	MIN	un	400	300	250	200	1,150
	MAX	un	800	800	400	400	2,400

Design the model so as to minimize transport costs.

```
MODEL:
SETS:
WAREHOUSES: DEM_MAX, DEM_MIN ;
PLANT: CAPACITY;
ROUTES( PLANT, WAREHOUSES): COST, VOLUME;
ENDSETS
DATA:
!  Warehouses attributes;
WAREHOUSES,     DEM_MIN,   DEM_MAX =
WA              400        800
WB              300        800
WC              250        800
WD              200        400;
! Plant attributes;
PLANT,          CAPACITY   =
P1              400
P2              500
P3              700;
! Cost values   WA    WB    WC    WD;
COST       =    2     5     4     8      ! P1;
                3     2     5     4      ! P2;
                5     3     9     5;     ! P3;
ENDDATA
SUBMODEL MIN8:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( WAREHOUSES( J):
        [DEM0] @SUM( PLANT( I): VOLUME( I, J)) >= DEM_MIN( J);
        [DEM1] @SUM( PLANT( I): VOLUME( I, J)) <= DEM_MAX( J););
! The capacity constraints;
@FOR( PLANT( I):
        [CAP] @SUM( WAREHOUSES( J): VOLUME( I, J)) <= CAPACITY( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA", @NEWLINE(1), "  SHIPPING COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE("  ", @NEWLINE( 1), "  AVAILABLE:", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE("  ", @NEWLINE( 1), "  MINIMUM DEMAND:",    @NEWLINE( 1));
@TABLE(DEM_MIN);
@WRITE("  ", @NEWLINE( 1), "  MAXIMUM DEMAND:",    @NEWLINE( 1));
@TABLE(DEM_MAX);
@WRITE("  ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MIN8);
! Solution Report;
@WRITE("  ", @NEWLINE(1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From: ',
        @FORMAT(PLANT( I),'-3s'), ' Ship to: ' ,
        @FORMAT(WAREHOUSES( J),'-3s'), ' ',
        @FORMAT(VOLUME( I, J),'%3.0f'),'Un x Cost: $',
        @FORMAT(COST( I, J), '%3.2f'),' = Total: $',
        @FORMAT(COST( I, J) * VOLUME(I,J),'%7.2f'),
@NEWLINE( 1));
!@GEN(MIN8);   ! To see the corresponding model scalar, remove (!);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA
SHIPPING COST:
           WA        WB        WC        WD
 P1  2.000000  5.000000  4.000000  8.000000
 P2  3.000000  2.000000  5.000000  4.000000
 P3  5.000000  3.000000  9.000000  5.000000

AVAILABLE:
 P1  400.0000
 P2  500.0000
 P3  700.0000

MINIMUM DEMAND:
 WA  400.0000
 WB  300.0000
 WC  250.0000
 WD  200.0000

MAXIMUM DEMAND:
 WA  800.0000
 WB  800.0000
 WC  800.0000
 WD  400.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              3700.000
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
From: P1  Ship to: WA  400Un x Cost: $2.00 = Total: $ 800.00
From: P2  Ship to: WB  250Un x Cost: $2.00 = Total: $ 500.00
From: P2  Ship to: WC  250Un x Cost: $5.00 = Total: $1250.00
From: P3  Ship to: WB   50Un x Cost: $3.00 = Total: $ 150.00
From: P3  Ship to: WD  200Un x Cost: $5.00 = Total: $1000.00
```

9

GOAL

A four-plant zipper company, whose production capacity, unit price of each product, demand of each distributor and the unit cost of transportation, are shown below:

Plant To Distributors		Unit Cost of Transport							Unit Cost	Capacity (Unit)
		D1	D2	D3	D4	D5	D6	D7		
P1	$	2.50	2.75	1.75	2.00	2.10	1.80	1.65	35.50	18,000
P2	$	1.85	1.90	1.50	1.60	1.00	1.90	1.85	37.50	15,000
P3	$	2.30	2.25	1.85	1.25	1.50	2.25	2.00	39.00	25,000
P4	$	1.90	0.90	1.60	1.75	2.00	2.50	2.65	36.25	20,000
Demand	un	8,500	14,500	13,500	12,600	18,000	15,000	9,000	-	-

The Company wants to determine the cheapest way to send its distributors from the various factories, in order to minimize cost.

As total demand exceeds the production capacity of all plants, it has been determined that at least 80% of each demand should be met.

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Distribution
- Logistics

Source:

- Book 2
- Page 242

```
MODEL:
SETS:
DISTRIBUTORS: DEMAND ;
PLANT: COST, CAPACITY;
ROUTES( PLANT, DISTRIBUTORS): SHIPPING_COST, VOLUME;
ENDSETS
DATA:
! Distributors attributes;
DISTRIBUTORS,     DEMAND =
D1                8500
D2                14500
D3                13500
D4                12600
D5                18000
D6                15000
D7                9000;

! Plant attributes;
PLANT,            COST,     CAPACITY   =
P1                35.50     18000
P2                37.50     15000
P3                39.00     25000
P4                36.25     20000;

! Shipping_Cost     D1    D2    D3    D4    D5    D6    D7;
SHIPPING_COST   =  2.50  2.75  1.75  2.00  2.10  1.80  1.65   ! P1;
                   1.85  1.90  1.50  1.60  1.00  1.90  1.85   ! P2;
                   2.30  2.25  1.85  1.25  1.50  2.25  2.00   ! P3;
                   1.90    0.90 1.60  1.75  2.00  2.50  2.65;  ! P4;
ENDDATA
SUBMODEL MAX9:
[OBJ] MIN = @SUM( ROUTES( I, J): (COST( I) + SHIPPING_COST(I,J)) * VOLUME( I, J));
! The demand constraints;
@FOR( DISTRIBUTORS( J):
        [DEM0] @SUM( PLANT( I): VOLUME( I, J)) >= DEMAND( J) * 0.80;);
! The capacity constraints;
@FOR( PLANT( I):
        [CAP] @SUM( DISTRIBUTORS( J): VOLUME( I, J)) <= CAPACITY( I));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA", @NEWLINE(1), "  SHIPPING COST:", @NEWLINE( 1));
@TABLE(SHIPPING_COST);
@WRITE("  ", @NEWLINE( 1), "  COST (un):", @NEWLINE( 1));
@TABLE(COST);
@WRITE("  ", @NEWLINE( 1), "  CAPACITY:", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE("  ", @NEWLINE( 1), "  DEMAND:",   @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE("  ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX9);
! Solution Report;
@WRITE("  ", @NEWLINE(1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 2));
@WRITE( "  ", "PRODUCTION COST: ", @NEWLINE(1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From: ',
        @FORMAT(PLANT( I),'-3s'), ' Ship to: ' ,
        @FORMAT(DISTRIBUTORS( J),'-3s'), ' ',
        @FORMAT(VOLUME( I, J),'%5.0f'),'un x Cost: $',
        @FORMAT(COST( I), '%3.2f'),' = Total: $',
        @FORMAT(COST( I) * VOLUME(I,J),'%10.2f'),
@NEWLINE( 1));
! Total;
@WRITE("  ", "Total:", 49*' ','$',
        @FORMAT(@SUM( ROUTES( I, J): COST( I) * VOLUME( I, J)),'%9.2f'),
@NEWLINE(2),);
@WRITE("  ", "SHIPPING COST: ", @NEWLINE(1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From: ',
        @FORMAT(PLANT( I),'-3s'), ' Ship to: ' ,
        @FORMAT(DISTRIBUTORS( J),'-3s'), ' ',
        @FORMAT(VOLUME( I, J),'%5.0f'),'un x Cost: $',
        @FORMAT(SHIPPING_COST( I,J), '%5.2f'),' = Total: $',
        @FORMAT(SHIPPING_COST( I,J) * VOLUME(I,J),'%10.2f'),
@NEWLINE( 1));
! Total;
@WRITE("  ", "Total:", 49*' ','$',
        @FORMAT(@SUM( ROUTES( I, J): SHIPPING_COST( I,J) * VOLUME( I, J)),'%10.2f'),
@NEWLINE(2));
! Total;
@WRITE("  ", "Total Cost:", 44*' ','$',
        @FORMAT(@SUM( ROUTES( I, J): (COST(I) + SHIPPING_COST( I,J)) * VOLUME( I, J)),'%10.2f'),
@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX9);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA
SHIPPING COST:
            D1         D2         D3         D4         D5         D6         D7
 P1   2.500000   2.750000   1.750000   2.000000   2.100000   1.800000   1.650000
 P2   1.850000   1.900000   1.500000   1.600000   1.000000   1.900000   1.850000
 P3   2.300000   2.250000   1.850000   1.250000   1.500000   2.250000   2.000000
 P4   1.900000  0.9000000   1.600000   1.750000   2.000000   2.500000   2.650000

COST (un):
 P1  35.50000
 P2  37.50000
 P3  39.00000
 P4  36.25000

CAPACITY:
 P1  18000.00
 P2  15000.00
 P3  25000.00
 P4  20000.00

DEMAND:
 D1  8500.000
 D2  14500.00
 D3  13500.00
 D4  12600.00
 D5  18000.00
 D6  15000.00
 D7  9000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                          2805450.00
Infeasibilities:                                          0.00000000

IDEAL TRANSPORT PROGRAM:

PRODUCTION COST:
From: P1  Ship to: D6  12000un x Cost: $35.50 = Total: $ 426000.00
From: P1  Ship to: D7   6000un x Cost: $35.50 = Total: $ 213000.00
From: P2  Ship to: D3    600un x Cost: $37.50 = Total: $  22500.00
From: P2  Ship to: D5  14400un x Cost: $37.50 = Total: $ 540000.00
From: P3  Ship to: D3   8600un x Cost: $39.00 = Total: $ 335400.00
From: P3  Ship to: D4  10080un x Cost: $39.00 = Total: $ 393120.00
From: P3  Ship to: D7   1200un x Cost: $39.00 = Total: $  46800.00
From: P4  Ship to: D1   6800un x Cost: $36.25 = Total: $ 246500.00
From: P4  Ship to: D2  11600un x Cost: $36.25 = Total: $ 420500.00
From: P4  Ship to: D3   1600un x Cost: $36.25 = Total: $  58000.00
Total:                                                   $2701820.00

SHIPPING COST:
From: P1  Ship to: D6  12000un x Cost: $ 1.80 = Total: $  21600.00
From: P1  Ship to: D7   6000un x Cost: $ 1.65 = Total: $   9900.00
From: P2  Ship to: D3    600un x Cost: $ 1.50 = Total: $    900.00
From: P2  Ship to: D5  14400un x Cost: $ 1.00 = Total: $  14400.00
From: P3  Ship to: D3   8600un x Cost: $ 1.85 = Total: $  15910.00
From: P3  Ship to: D4  10080un x Cost: $ 1.25 = Total: $  12600.00
From: P3  Ship to: D7   1200un x Cost: $ 2.00 = Total: $   2400.00
From: P4  Ship to: D1   6800un x Cost: $ 1.90 = Total: $  12920.00
From: P4  Ship to: D2  11600un x Cost: $ 0.90 = Total: $  10440.00
From: P4  Ship to: D3   1600un x Cost: $ 1.60 = Total: $   2560.00
Total:                                                   $ 103630.00

Total Cost:                                              $2805450.00
```

10

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Distribution
- Logistics

Source:

- Book 4
- WIDGET

GOAL

Suppose the Wireless Widget (WW) company has six warehouses that provide eight vendors with

Warehouses / Vendor		V1	V2	V3	V4	V5	V6	V7	V8	Available (un)
WH1	$	6	2	6	7	4	2	5	9	60
WH2	$	4	9	5	3	8	5	8	2	55
WH3	$	5	2	1	9	7	4	3	3	51
WH4	$	7	6	7	3	9	2	7	1	43
WH5	$	2	3	9	5	7	2	6	5	41
WH6	$	5	5	2	2	8	1	4	3	52
Demand	un	35	37	22	32	41	32	43	38	-

their widgets. All the data for the development of the model are described below:

Each warehouse has a set of widgets that can't be exceeded and each vendor has demand for widgets that must be satisfied.

WW wants to determine how many widgets to send from each warehouse to each vendor in order to minimize the total cost of shipping.

```
MODEL:
SETS:
WAREHOUSES: CAPACITY;
VENDORS: DEMAND;
ROUTES( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
DATA:
!  Warehouses attributes;
WAREHOUSES,        CAPACITY =
WH1                60
WH2                55
WH3                51
WH4                43
WH5                41
WH6                52;
! Vendor attributes;
VENDORS,           DEMAND   =
V1                 35
V2                 37
V3                 22
V4                 32
V5                 41
V6                 32
V7                 43
V8                 38;
! Cost values      V1    V2    V3    V4    V5    V6    V7    V8;
COST          =    6     2     6     7     4     2     5     9      ! WH1;
                   4     9     5     3     8     5     8     2      ! WH2;
                   5     2     1     9     7     4     3     3      ! WH3;
                   7     6     7     3     9     2     7     1      ! WH4;
                   2     3     9     5     7     2     6     5      ! WH5;
                   5     5     2     2     8     1     4     3;     ! WH6;
ENDDATA
SUBMODEL MIN10:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J) * VOLUME( I, J));
@FOR( VENDORS( J):                                                  ! The demand constraints;
        [DEM] @SUM( WAREHOUSES( I): VOLUME( I, J)) = DEMAND( J));
@FOR( WAREHOUSES( I):                                               ! The capacity constraints;
         [CAP] @SUM( VENDORS( J): VOLUME( I, J)) <= CAPACITY( I));
ENDSUBMODEL
CALC:
@SET('TERSEO',1);                                                   ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);                                                   ! Post status windows, 1 Yes, 0 No;
@WRITE("  DATA", @NEWLINE(1), "  SHIPPING COST:", @NEWLINE( 1));    ! Data Block;
@TABLE(COST);
@WRITE("  ", @NEWLINE( 1), "AVAILABLE:", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE("  ", @NEWLINE( 1), "DEMAND:",    @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE("  ", @NEWLINE( 1), "SOLUTION ", @NEWLINE( 1));
@SOLVE(MIN10);                                                            ! Execute sub-model;
@WRITE("  ", @NEWLINE(1), "IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));     ! Solution Report;
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From: ',
        @FORMAT(WAREHOUSES( I),'-3s'), '   Ship to: ' ,
        @FORMAT(VENDORS( J),'-3s'), '   ',
        @FORMAT(VOLUME( I, J),'%2.0f'),'Un   x   Unit cost: $',
        @FORMAT(COST( I, J), '%3.2f'),'  =  Total: $',
        @FORMAT(COST( I, J) * VOLUME(I,J),'%6.2f'), @NEWLINE( 1));
!@GEN(MIN10);                                                        ! To see the corresponding model scalar, remove (!);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING COST:
          V1        V2        V3        V4        V5        V6        V7        V8
 WH1  6.000000  2.000000  6.000000  7.000000  4.000000  2.000000  5.000000  9.000000
 WH2  4.000000  9.000000  5.000000  3.000000  8.000000  5.000000  8.000000  2.000000
 WH3  5.000000  2.000000  1.000000  9.000000  7.000000  4.000000  3.000000  3.000000
 WH4  7.000000  6.000000  7.000000  3.000000  9.000000  2.000000  7.000000  1.000000
 WH5  2.000000  3.000000  9.000000  5.000000  7.000000  2.000000  6.000000  5.000000
 WH6  5.000000  5.000000  2.000000  2.000000  8.000000  1.000000  4.000000  3.000000

AVAILABLE (un):
 WH1  60.00000
 WH2  55.00000
 WH3  51.00000
 WH4  43.00000
 WH5  41.00000
 WH6  52.00000

DEMAND (un):
 V1  35.00000
 V2  37.00000
 V3  22.00000
 V4  32.00000
 V5  41.00000
 V6  32.00000
 V7  43.00000
 V8  38.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                                                    664.0000
Infeasibilities:                                                    0.000000

IDEAL TRANSPORT PROGRAM:
From: WH1   Ship to: V2    19Un   x   Unit cost: $2.00  =  Total: $ 38.00
From: WH1   Ship to: V5    41Un   x   Unit cost: $4.00  =  Total: $164.00
From: WH2   Ship to: V1     1Un   x   Unit cost: $4.00  =  Total: $  4.00
From: WH2   Ship to: V4    32Un   x   Unit cost: $3.00  =  Total: $ 96.00
From: WH3   Ship to: V2    11Un   x   Unit cost: $2.00  =  Total: $ 22.00
From: WH3   Ship to: V7    40Un   x   Unit cost: $3.00  =  Total: $120.00
From: WH4   Ship to: V6     5Un   x   Unit cost: $2.00  =  Total: $ 10.00
From: WH4   Ship to: V8    38Un   x   Unit cost: $1.00  =  Total: $ 38.00
From: WH5   Ship to: V1    34Un   x   Unit cost: $2.00  =  Total: $ 68.00
From: WH5   Ship to: V2     7Un   x   Unit cost: $3.00  =  Total: $ 21.00
From: WH6   Ship to: V3    22Un   x   Unit cost: $2.00  =  Total: $ 44.00
From: WH6   Ship to: V6    27Un   x   Unit cost: $1.00  =  Total: $ 27.00
From: WH6   Ship to: V7     3Un   x   Unit cost: $4.00  =  Total: $ 12.00
```

11

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Warehouse
- Distribution
- Consumer
- Logistics

Source:

- Book 6
- Page 836

GOAL

Minimize shipping costs in a three-tiered distribution system involves plants, distribution centers and customers.

The plants produce multiple products, which are sent to distribution centers. If a distribution center is used, it incurs a fixed cost.

Plant / Distribution Center			Shipping Cost			
Products	Plant	Capacity	DC1	DC2	DC3	DC4
PA	P1	80	1.00	3.00	3.00	5.00
	P2	40	4.00	4.50	1.50	3.80
	P3	75	2.00	3.30	2.20	3.20
PB	P1	20	1.00	2.00	2.00	5.00
	P2	60	4.00	4.60	1.30	3.50
	P3	75	1.80	3.00	2.00	3.50
DC Fixed Cost			100.00	150.00	160.00	139.00

Customers are provided by a single distribution center. Four DC1, DC2, DC3 and DC4 distribution centers Intermediate receipt of production and deliver to customers and cost of shipping to consumers in tons. Fixed cost for each DC that delivers deliveries.

Each consumer has a specific demand and will be served by a single DC.

SHIPPING COST											
Distribution Center		DC To C (PA)					DC To C (PB)				
		C1	C2	C3	C4	C5	C1	C2	C3	C4	C5
DC1	$	5.00	5.00	3.00	2.00	4.00	5.00	4.90	3.30	2.50	4.10
DC2	$	5.10	4.90	3.30	2.50	2.70	5.00	4.80	3.00	2.20	2.50
DC3	$	3.50	2.00	1.90	4.00	4.30	3.20	2.00	1.70	3.50	4.00
DC4	$	1.00	1.80	4.90	4.80	2.00	1.50	2.00	5.00	5.00	2.30
Demand	ton	25	30	50	15	35	25	8	0	30	30

This model applies to any multi-level solution as long as the data structure meets;

```
MODEL:
SETS:
 PRODUCT/ A, B/;                              ! Two products;
 PLANT/ P1, P2, P3/;                          ! Three plants;
 DISTCTR/ DC1, DC2, DC3, DC4/: F, Z;          ! Each DC has an associated fixed cost, F, and an "open" indicator, Z.;
 CUSTOMER/ C1, C2, C3, C4, C5/;               ! Five customers;
 DEMLINK( PRODUCT, CUSTOMER): D;              ! D = Demand for a product by a customer.;
 SUPLINK( PRODUCT, PLANT): S;                 ! S = Capacity for a product at a plant.;
 YLINK( DISTCTR, CUSTOMER): Y;                ! Each customer is served by one DC, indicated by Y.;
 CLINK( PRODUCT, PLANT, DISTCTR): C, X;       ! C = cost/ton of a product from a plant to a DC, X=tons shipped.;
 GLINK( PRODUCT, DISTCTR, CUSTOMER): G;       ! G = cost/ton of a product from a DC to a customer.;
ENDSETS
DATA:
! Plant Capacities              P1     P2     P3;
S =                             80,    40,    75,              ! Product: A;
                                20,    60,    75;              ! Product: B;
! Shipping cost  - Plant to DC  DC1    DC2    DC3    DC4;
C =                             1,     3,     3,     5,        ! Product A;
                                4,     4.5,   1.5,   3.8,
                                2,     3.3,   2.2,   3.2,
                                1,     2,     2,     5,        ! Product B;
                                4,     4.6,   1.3,   3.5,
                                1.8,   3,     2,     3.5;

! DC Fixed cost                 DC1    DC2    DC3    DC4;
F =                             100,   150,   160,   139;
! Shipping cost - DC to customer C1    C2     C3     C4     C5;
G =                             5,     5,     3,     2,     4,      ! Product A;
                                5.1,   4.9,   3.3,   2.5,   2.7,
                                3.5,   2,     1.9,   4,     4.3,
                                1,     1.8,   4.9,   4.8,   2,
                                5,     4.9,   3.3,   2.5,   4.1,    ! Product B;
                                5,     4.8,   3,     2.2,   2.5,
                                3.2,   2,     1.7,   3.5,   4,
                                1.5,   2,     5,     5,     2.3;
! Customer Demands              C1     C2     C3     C4     C5;
D =                             25,    30,    50,    15,    35,
                                25,    8,     0,     30,    30;
ENDDATA
SUBMODEL MIN11:
[OBJ] MIN =   SHIPDC + SHIPCOST + FXCOST;
              [SHPDC] SHIPDC     = @SUM( CLINK: C * X);
              [SHCUS] SHIPCOST   = @SUM( GLINK( I, K, L): G( I, K, L) * D( I, L) * Y( K, L));
              [FCOS]  FXCOST     = @SUM( DISTCTR: F * Z);
! Supply Constraints;
@FOR( PRODUCT( I): @FOR( PLANT( J):
        [SUP] @SUM( DISTCTR( K): X( I, J, K)) <= S( I, J)));
! DC balance constraints;
@FOR( PRODUCT( I): @FOR( DISTCTR( K):
        [BAL] @SUM( PLANT( J): X( I, J, K)) = @SUM( CUSTOMER( L): D( I, L)* Y( K, L))));
! Demand constraints;
@FOR( CUSTOMER( L):
        [DEM] @SUM( DISTCTR( K): Y( K, L)) = 1);
! Force DC K open if it serves customer L;
@FOR( CUSTOMER( L):
      @FOR( DISTCTR( K): [DCK]  Y( K, L) <= Z( K)));
! Y Binary;
@FOR( DISTCTR( K):
       @FOR( CUSTOMER( L): [BIN] @BIN( Y( K,L))));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Fixed line length
@SET('LINLEN',120);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  Shipping cost, Plant to Distribution Center ($/ton):", @NEWLINE( 1));
@TABLE(C);
@WRITE(" ", @NEWLINE( 1), "  Plant capacity (ton):", @NEWLINE( 1));
@TABLE(S);
@WRITE(" ", @NEWLINE( 1), "  Consumer demand (ton):", @NEWLINE( 1));
@TABLE(D);
@WRITE(" ", @NEWLINE( 1), "  Fixed cost distribution Center:", @NEWLINE( 1));
@TABLE(F);
@WRITE(" ", @NEWLINE( 1));
@WRITE("  Shipping cost, Distribution Center to Consumer ($/ton): ", @NEWLINE( 1));
@TABLE(G);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION ", @NEWLINE( 1), " ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN11);
! Solution report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITE(" PRODUCT/PLANT TO DISTRIBUTION CENTER ", @NEWLINE(1));
@WRITEFOR( CLINK( I, K, L)| C( I, K, L) * X( i, K, L ) #GT# 0:
     ' Product: ', PRODUCT(I),'  ',' Plant: ', PLANT(I), ' ',' TO  Distctr: ', DISTCTR(L), ' ', ' Shipping cost: $',
     @FORMAT(C( I, K, L) * X( i, K, L ) ,'%6.2f'),
@NEWLINE( 1));
@WRITE(' Total:',51*' ','$',@FORMAT(SHIPDC,'%3.2f'),@NEWLINE( 2));
@WRITE(" DISTRIBUTION CENTER FIXED COST ", @NEWLINE(1));
@WRITEFOR( CLINK( I, K, L) |  C( I, K, L) * X( i, K, L ) #GT# 0:
     ' Product: ', Product(I), '   Plant: ', Plant(I), '  TO  Distctr: ', DISTCTR(L),
@NEWLINE( 1));
@WRITE(' Total:',50*' ',' $',@FORMAT(FXCOST,'%3.2f'),@NEWLINE( 2));
@WRITE(" DISTRIBUTION CENTER TO CONSUMER ", @NEWLINE(1));
@WRITEFOR( GLINK( I, K, L)| G( I, K, L) * D( I, L) * Y( K, L) #GT# 0:
     ' Product: ', PRODUCT(I),' Distctr: ', DISTCTR(K), ' TO',' Customer: ',CUSTOMER(L), ' ', '  Shipping cost: $',
     @FORMAT(G( I, K, L) * D( I, L) * Y( K, L),'%6.2f'),
@NEWLINE( 1));
@WRITE(' Total:',50*' ',' $',@FORMAT(SHIPCOST,'%3.2f'),@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN11);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
Shipping cost, Planta to Distribution Center ($/ton):
              DC1       DC2       DC3       DC4
 A    P1  1.000000  3.000000  3.000000  5.000000
 A    P2  4.000000  4.500000  1.500000  3.800000
 A    P3  2.000000  3.300000  2.200000  3.200000
 B    P1  1.000000  2.000000  2.000000  5.000000
 B    P2  4.000000  4.600000  1.300000  3.500000
 B    P3  1.800000  3.000000  2.000000  3.500000

Plant capacity (ton):
         P1        P2        P3
 A  80.00000  40.00000  75.00000
 B  20.00000  60.00000  75.00000

Consumer demand (ton):
         C1        C2        C3        C4        C5
 A  25.00000  30.00000  50.00000  15.00000  35.00000
 B  25.00000  8.000000  0.000000  30.00000  30.00000

Fixed cost distribution Center:
 DC1  100.0000
 DC2  150.0000
 DC3  160.0000
 DC4  139.0000

Shipping cost, Distribution Center to Consumer ($/ton):
                 C1        C2        C3        C4        C5
 A    DC1  5.000000  5.000000  3.000000  2.000000  4.000000
 A    DC2  5.100000  4.900000  3.300000  2.500000  2.700000
 A    DC3  3.500000  2.000000  1.900000  4.000000  4.300000
 A    DC4  1.000000  1.800000  4.900000  4.800000  2.000000
 B    DC1  5.000000  4.900000  3.300000  2.500000  4.100000
 B    DC2  5.000000  4.800000  3.000000  2.200000  2.500000
 B    DC3  3.200000  2.000000  1.700000  3.500000  4.000000
 B    DC4  1.500000  2.000000  5.000000  5.000000  2.300000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
 Global optimal solution found.
 Objective value:                              1354.400
 Objective bound:                              1354.400
 Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
PRODUCT/PLANT TO DISTRIBUTION CENTER
Product: A   Plant: P1  TO  Distctr: DC1  Shipping cost: $ 50.00
Product: A   Plant: P1  TO  Distctr: DC3  Shipping cost: $ 60.00
Product: A   Plant: P1  TO  Distctr: DC3  Shipping cost: $143.00
Product: B   Plant: P2  TO  Distctr: DC1  Shipping cost: $ 20.00
Product: B   Plant: P2  TO  Distctr: DC3  Shipping cost: $ 42.90
Product: B   Plant: P2  TO  Distctr: DC1  Shipping cost: $ 72.00
Total:                                                   $387.90

DISTRIBUTION CENTER FIXED COST
Product: A   Plant: P1  TO  Distctr: DC1
Product: A   Plant: P1  TO  Distctr: DC3
Product: A   Plant: P1  TO  Distctr: DC3
Product: B   Plant: P2  TO  Distctr: DC1
Product: B   Plant: P2  TO  Distctr: DC3
Product: B   Plant: P2  TO  Distctr: DC1
Total:                                                   $260.00

DISTRIBUTION CENTER TO CONSUMER
Product: A Distctr: DC1 TO Customer: C4   Shipping cost: $ 30.00
Product: A Distctr: DC1 TO Customer: C5   Shipping cost: $140.00
Product: A Distctr: DC3 TO Customer: C1   Shipping cost: $ 87.50
Product: A Distctr: DC3 TO Customer: C2   Shipping cost: $ 60.00
Product: A Distctr: DC3 TO Customer: C3   Shipping cost: $ 95.00
Product: B Distctr: DC1 TO Customer: C4   Shipping cost: $ 75.00
Product: B Distctr: DC1 TO Customer: C5   Shipping cost: $123.00
Product: B Distctr: DC3 TO Customer: C1   Shipping cost: $ 80.00
Product: B Distctr: DC3 TO Customer: C2   Shipping cost: $ 16.00
Total:                                                   $706.50
```

12

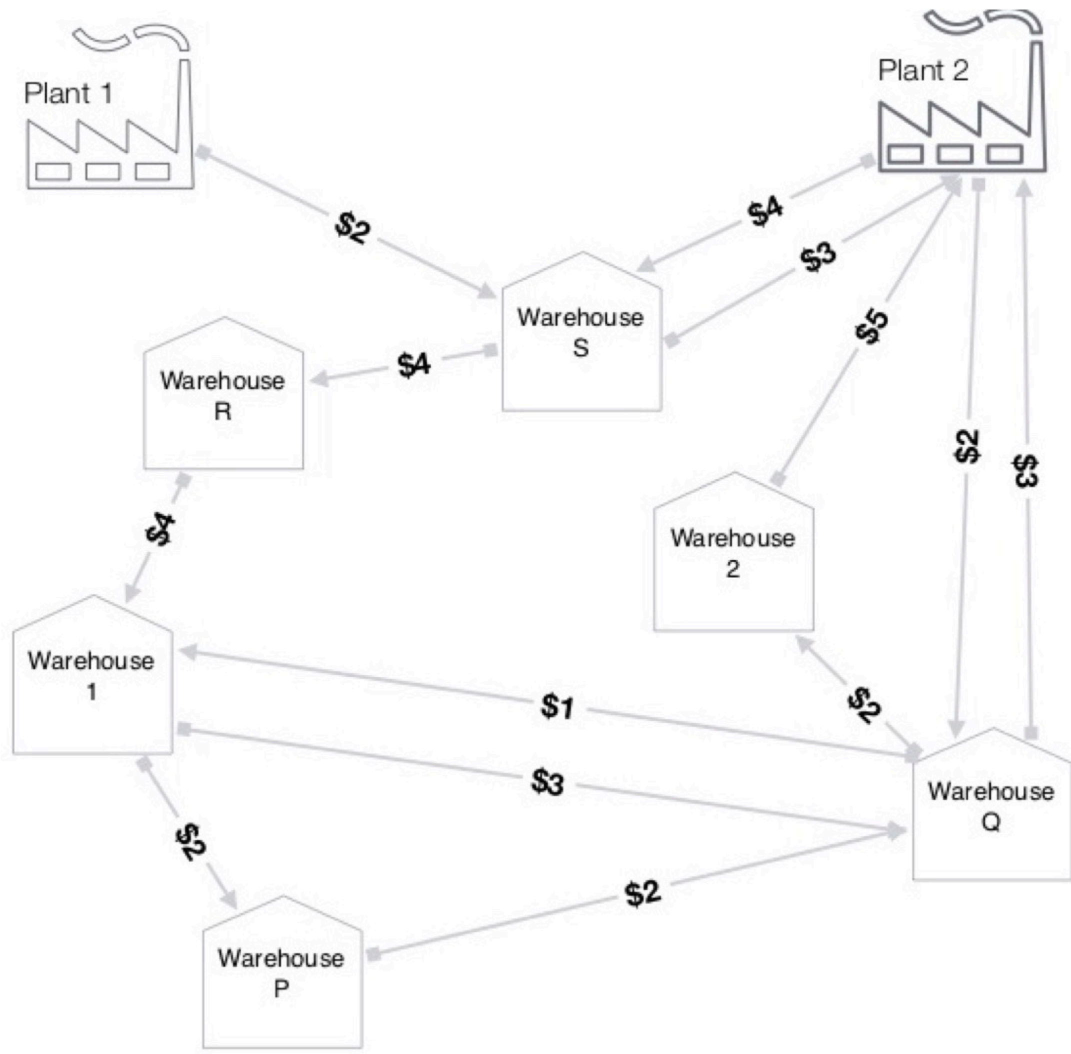
Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Delivery
- Warehouse
- Distribution
- Logistics

Source:
- Book 6
- Page 836

GOAL

The simplified problem of transport always has that a trajectory starts at an origin and there is a single path to a destination. In the advanced problem we can have any composition of trajectories.



The mass balance at any intermediate station should be zero. Build a model to get the lowest cost of transportation that meets all requirements.

Routes / Cost		To	WS	WR	WQ	WP	W1	W2	P1	P2
From	P1	$	2.00							
	P2	$	4.00		2.00			5.00		
	W1	$			3.00	2.00				
	W2	$								
	WS	$		4.00						3.00
	WR	$					4.00			
	WQ	$					1.00	2.00		3.00
	WP	$			2.00					
Demand		un					15	25		
Maximum Production									40	15

```
MODEL:
SETS:
ROUTE:COST, VOL;
PLANT: MAX_PROD;
WAREHOUSE:DEMAND;
ENDSETS
DATA:
! Routes attributes;
ROUTE,        COST =
P1_WS         2     ! VOL(1)     Plant 1              to Warehouse S;
WS_WR         4     ! VOL(2)     Warehouse S  to Warehouse R;
WS_P2         3     ! VOL(3)     Warehouse S  to Plant 2;
WR_W1         4     ! VOL(4)     Warehouse R  to Warehouse 1;
W1_WP         2     ! VOL(5)     Warehouse 1  to Warehouse P;
W1_WQ         3     ! VOL(6)     Warehouse 1  to Warehouse Q;
WP_WQ         2     ! VOL(7)     Warehouse P  to Warehouse Q;
WQ_W1         1     ! VOL(8)     Warehouse Q  to Warehouse 1;
WQ_W2         2     ! VOL(9)     Warehouse Q  to Warehouse 2;
WQ_P2         3     ! VOL(10)    Warehouse Q  to Plant 2;
P2_WQ         2     ! VOL(11)    Plant 2      to Warehouse Q;
P2_W2         5     ! VOL(12)    Plant 2      to Warehouse 2;
P2_WS         4;    ! VOL(13)    Plant 2      to Warehouse S;
! Plants attributes;
PLANT,        MAX_PROD =
P1            40
P2            15;
! Warehouse attributes;
WAREHOUSE,    DEMAND =
W1            15
W2            25;
ENDDATA
SUBMODEL MIN12:
[OBJ] MIN = @SUM( ROUTE( I): COST( I) * VOL( I));
! Factory Production Restriction
VOL(1) <= MAX_PROD(1);
VOL(13) + VOL(11) + VOL(12) - VOL(3) - VOL(10) <= MAX_PROD(2);
! Demand Restriction ;
VOL(4) + VOL(8) - VOL(6) - VOL(5) = DEMAND(1);
VOL(9) + VOL(12) = DEMAND(2);
! Balance ;                                   ;
VOL(1) + VOL(13) - VOL(2) - VOL(3) = 0;
VOL(2) - VOL(4) = 0;
VOL(5) - VOL(7) = 0;
VOL(6) + VOL(11) + VOL(7) - VOL(8) - VOL(10) - VOL(9) = 0;
ENDSUBMODEL
CALC:
@SET('TERSEO',1);         ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);         ! Post status windows, 1 Yes, 0 No;
@WRITE("  DATA", @NEWLINE(1),"  SHIPPING COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE("  ", @NEWLINE( 1), "  PRODUCTION (un):", @NEWLINE( 1));
@TABLE(MAX_PROD);
@WRITE("  ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE("  ", @NEWLINE( 1), "  SOLUTION", @NEWLINE( 1));
@SOLVE(MIN12);            ! Execute sub-model;
@WRITE("  ", @NEWLINE(1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));       ! Solution Report;
@WRITEFOR( ROUTE( I)| VOL(I) #GT# 0:'  Route: ',
      @FORMAT(ROUTE( I),'-3s'), ' Ship: ',
      @FORMAT(VOL( I),'%2.0f'),'un x Cost: $',
      @FORMAT(COST( I), '%3.2f'),' = Total: $',
      @FORMAT(COST( I) * VOL(I),'%6.2f'),
@NEWLINE( 1));
!@GEN(MIN12);                  !To see the corresponding model scalar, remove (!);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA
SHIPPING COST:
 P1_WS  2.000000
 WS_WR  4.000000
 WS_P2  3.000000
 WR_W1  4.000000
 W1_WP  2.000000
 W1_WQ  3.000000
 WP_WQ  2.000000
 WQ_W1  1.000000
 WQ_W2  2.000000
 WQ_P2  3.000000
 P2_WQ  2.000000
 P2_W2  5.000000
 P2_WS  4.000000

PRODUCTION (un):
 P1  40.00000
 P2  15.00000

DEMAND (un):
 W1  15.00000
 W2  25.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              270.0000
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
Route: P1_WS Ship: 25un x Cost: $2.00 = Total: $ 50.00
Route: WS_P2 Ship: 25un x Cost: $3.00 = Total: $ 75.00
Route: WQ_W1 Ship: 15un x Cost: $1.00 = Total: $ 15.00
Route: WQ_W2 Ship: 25un x Cost: $2.00 = Total: $ 50.00
Route: P2_WQ Ship: 40un x Cost: $2.00 = Total: $ 80.00
```

13

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Delivery
- Distribution
- Supplier
- Logistics

Source:

- Book 2
- Page 228

GOAL

A network of building material stores has four stores that must be supplied with: This sand can be loaded into three ports PORT1, PORT2 and PORT3, whose distances (in km) can be seen below.

Port / Shop	To	S1	S2	S3	S4	Available (m3)
Port-1	km	30	20	24	18	1,000
Port-2	km	12	36	30	24	1,000
Port-3	km	8	15	25	20	1,000
Demand	m3	50	80	40	100	

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): DISTANCE, VOLUME;
ENDSETS
DATA:
! Available attributes;
SUPPLIER,          AVAILABLE   =
PORT1              1000
PORT2              1000
PORT3              1000;

!  Consumer attributes;
CONSUMER,          DEMAND        =
SHOP1              50
SHOP2              80
SHOP3              40
SHOP4              100;

! Distance         SHOP1      SHOP2      SHOP3      SHOP4;
DISTANCE   =       30         20         24         18         ! PORT1;
                   12         36         30         24         ! PORT2;
                   8          15         25         20;        ! PORT3;
ENDDATA
SUBMODEL MIN13:
[OBJ] MIN = @SUM( ROUTES( I, J): DISTANCE( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
        [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The available constraints;
@FOR( SUPPLIER( I):
        [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1),   "  SHIPPING DISTANCE ( km )", @NEWLINE( 1));
@TABLE(DISTANCE);
@WRITE(" ", @NEWLINE( 1),          "  AVAILABLE ( m3 )", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1),          "  DEMAND ( m3 ):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1),          "  NOTE: A Truck can carry 10m3 per trip.",
@NEWLINE( 2), "  SOLUTION ", @NEWLINE( 1));
@SOLVE(MIN13);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  Ship: ',
        @FORMAT(VOLUME( I, J),'%3.0f'), 'm3  From: ',
        @FORMAT(SUPPLIER( I),'-6s'),' To: ',
        @FORMAT(CONSUMER( J),'-6s'),'  Traveled: ',
        @FORMAT(DISTANCE (I, J) * VOLUME(I,J),'%4g'),'km', @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN11);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING DISTANCE ( km )
           SHOP1     SHOP2     SHOP3     SHOP4
 PORT1  30.00000  20.00000  24.00000  18.00000
 PORT2  12.00000  36.00000  30.00000  24.00000
 PORT3  8.000000  15.00000  25.00000  20.00000

AVAILABLE ( m3 )
 PORT1  1000.000
 PORT2  1000.000
 PORT3  1000.000

DEMAND ( m3 ):
 SHOP1  50.00000
 SHOP2  80.00000
 SHOP3  40.00000
 SHOP4  100.0000

NOTE: A Truck can carry 10m3 per trip.
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              4360.000
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
Ship:  40m3  From: PORT1  To: SHOP3   Traveled:  960km
Ship: 100m3  From: PORT1  To: SHOP4   Traveled: 1800km
Ship:  50m3  From: PORT3  To: SHOP1   Traveled:  400km
Ship:  80m3  From: PORT3  To: SHOP2   Traveled: 1200km
```

14

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Delivery
- Distribution
- Supplier
- Logistics

Source:
- Book 2
- Page 202

GOAL

Four gas stations SA, SB, SC and SD require 50, 40, 60 and 40 thousand gallons of gas, respectively.

It is possible to meet these demands through distributors D1, D2 and D3 which have 80, 100 and 50 thousand liters, respectively.

The costs of shipping 1 thousand gallons of gasoline are shown in the table below:

		Cost of shipping for every 1,000 gallons				Available (gal)
Distributor To		Station A	Station B	Station C	Station D	
D1	$	70.00	60.00	60.00	60.00	80,000
D2	$	50.00	80.00	60.00	70.00	100,000
D3	$	80.00	50.00	80.00	60.00	50,000
Demand	gal	50,000	40,000	60,000	40,000	-

It is possible to meet these demands through distributors D1, D2 and D3 which has availability of 80, 100 and 50 thousand liters, respectively.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;
CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): COST, VOLUME;
ENDSETS
DATA:
! Supplier attributes;
SUPPLIER,           AVAILABLE  =
D1                  80000
D2                  100000
D3                  50000;
!  Consumer attributes;
CONSUMER,           DEMAND      =
S1                  50000
S2                  40000
S3                  60000
S4                  40000;
! Cost              SA     SB     SC     SD;
COST   =            70     60     60     60      ! Distributor 1;
                    50     80     60     70      ! Distributor 2;
                    80     50     80     60;     ! Distributor 3;
ENDDATA
SUBMODEL MIN14:
[OBJ] MIN = @SUM( ROUTES( I, J): COST( I, J)/1000 * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
        [DEM] @SUM( SUPPLIER( I): VOLUME( I, J)) = DEMAND( J));
! The available constraints;
@FOR( SUPPLIER( I):
        [AVA] @SUM( CONSUMER( J): VOLUME( I, J)) <= AVAILABLE( I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE(" DATA:", @NEWLINE( 1),    " COST (for every 1000 gal):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1),           "  AVAILABLE ( gal )", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1),           "  DEMAND ( gal ):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(@NEWLINE( 2), "  SOLUTION ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN14);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0:'  From:',
        @FORMAT(SUPPLIER( I),'-2s'),' To:',
        @FORMAT(CONSUMER( J),'-2s')," ",
        @FORMAT(VOLUME( I, J),'%3.0f'), 'gal x Cost: $',
        @FORMAT(COST(I,J)/1000, '%4.2f'), ' = Total: $',
        @FORMAT(COST (I, J)/1000 * VOLUME(I,J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN14);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COST (for every 1000 gal):
            S1        S2        S3        S4
  D1  70.00000  60.00000  60.00000  60.00000
  D2  50.00000  80.00000  60.00000  70.00000
  D3  80.00000  50.00000  80.00000  60.00000

 AVAILABLE ( gal )
  D1  80000.00
  D2  100000.0
  D3  50000.00

 DEMAND ( gal ):
  S1  50000.00
  S2  40000.00
  S3  60000.00
  S4  40000.00
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              10500.00
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
From:D1 To:S3 40000gal x Cost: $0.06 = Total: $2400.00
From:D1 To:S4 40000gal x Cost: $0.06 = Total: $2400.00
From:D2 To:S1 50000gal x Cost: $0.05 = Total: $2500.00
From:D2 To:S3 20000gal x Cost: $0.06 = Total: $1200.00
From:D3 To:S2 40000gal x Cost: $0.05 = Total: $2000.00
```

15

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Delivery
- Consumer
- Supplier
- Logistics

Source:
- Book 2
- Page 238

GOAL

The removal of garbage from a large Brazilian City is an important and cost activity. Garbage needs to be collected from streets, sidewalks, industries, residences, etc., and arranged in appropriate places.

To enable rapid and efficient collection, the city has been divided into several sectors.

The garbage is collected and sent to dumps or to companies that do the waste treatment in order to take advantage of materials such as plastic and aluminum for recycling and produce fertilizer.

The capacity of each of the dumps, as well as the distance that the trucks accurately travel from the collection sectors to the delivery, besides the annual collection estimates, whose distances (in km) can be seen in the data block.

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

Sectors / Destiny			D1	D2	D3	D4	D5	Available
Sectors	Distance Between Origin/ Destination							1000/m3
	S1	Km	3.4	1.4	4.9	7.4	9.3	153
	S2	Km	2.4	2.1	8.3	9.1	8.8	152
	S3	Km	1.4	2.9	3.7	9.4	8.6	154
	S4	Km	2.6	3.6	4.5	8.2	8.9	138
	S5	Km	1.5	3.1	2.1	7.9	8.8	127
	S6	Km	4.2	4.9	6.5	7.7	6.1	129
	S7	Km	4.8	6.2	9.9	6.2	5.7	111
	S8	Km	5.4	6.0	5.2	7.6	4.9	110
	S9	Km	3.1	4.1	6.6	7.5	7.2	130
	S10	Km	3.2	6.5	7.1	6.0	8.3	135
Demand		1000/m3	350	250	500	400	200	-

Construct a model that minimizes the total distance to be covered by the trucks.

```
MODEL:
SETS:
SUPPLIER: AVAILABLE;   CONSUMER: DEMAND;
ROUTES( SUPPLIER, CONSUMER): DISTANCE, VOLUME;
ENDSETS
DATA:
!Supplier attributes;
SUPPLIER ,          AVAILABLE =
S1                  153
S2                  152
S3                  154
S4                  138
S5                  127
S6                  129
S7                  111
S8                  110
S9                  130
S10                 135;

! Consumer attributes;
CONSUMER ,          DEMAND =
D1                  350
D2                  250
D3                  500
D4                  400
D5                  200;

! Distance          D1    D2    D3    D4    D5;
DISTANCE  =         3.4   1.4   4.9   7.4   9.3    ! S1;
                    2.4   2.1   8.3   9.1   8.8    ! S2;
                    1.4   2.9   3.7   9.4   8.6    ! S3;
                    2.6   3.6   4.5   8.2   8.9    ! S4;
                    1.5   3.1   2.1   7.9   8.8    ! S5;
                    4.2   4.9   6.5   7.7   6.1    ! S6;
                    4.8   6.2   9.9   6.2   5.7    ! S7;
                    5.4   6.0   5.2   7.6   4.9    ! S8;
                    3.1   4.1   6.6   7.5   7.2    ! S9;
                    3.2   6.5   7.1   6.0   8.3;   ! S10;
ENDDATA
SUBMODEL MIN15:
[OBJ] MIN = @SUM( ROUTES( I, J): DISTANCE( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( CONSUMER( J):
        [DEM] @SUM( SUPPLIER( I): VOLUME( I, J))   <= DEMAND( J));
! The capacity constraints;
@FOR( SUPPLIER( I):
        [AVA]  @SUM( CONSUMER( J): VOLUME( I, J)) = AVAILABLE( I));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  SHIPPING DISTANCE ( km )", @NEWLINE( 1));
@TABLE(DISTANCE);
@WRITE(" ", @NEWLINE( 1),       "  AVAILABLE ( 1000/m3 )",      @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1),       "  DEMAND ( 1000/m3 ):",        @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1),         "  SOLUTION ",                      @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN15);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL TRANSPORT PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES( I, J) | VOLUME( I, J) #GT# 0: '  Ship ',
      @FORMAT(VOLUME( I, J),'%5.1f'), 'm3    From:',
      @FORMAT(SUPPLIER( I),'3s'), '    To:',
      @FORMAT(CONSUMER( J),'3s'), '   Traveled:',
      @FORMAT(DISTANCE (I, J) * VOLUME(I,J),'6.1f'),'km',
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
! @GEN(MIN11);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
SHIPPING DISTANCE ( km )
            D1        D2        D3        D4        D5
 S1   3.400000  1.400000  4.900000  7.400000  9.300000
 S2   2.400000  2.100000  8.300000  9.100000  8.800000
 S3   1.400000  2.900000  3.700000  9.400000  8.600000
 S4   2.600000  3.600000  4.500000  8.200000  8.900000
 S5   1.500000  3.100000  2.100000  7.900000  8.800000
 S6   4.200000  4.900000  6.500000  7.700000  6.100000
 S7   4.800000  6.200000  9.900000  6.200000  5.700000
 S8   5.400000  6.000000  5.200000  7.600000  4.900000
 S9   3.100000  4.100000  6.600000  7.500000  7.200000
 S10  3.200000  6.500000  7.100000  6.000000  8.300000

AVAILABLE ( 1000/m3 )
 S1   153.0000
 S2   152.0000
 S3   154.0000
 S4   138.0000
 S5   127.0000
 S6   129.0000
 S7   111.0000
 S8   110.0000
 S9   130.0000
 S10  135.0000

DEMAND ( 1000/m3 ):
 D1  350.0000
 D2  250.0000
 D3  500.0000
 D4  400.0000
 D5  200.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION
Global optimal solution found.
Objective value:                              4784.900
Infeasibilities:                              0.000000

IDEAL TRANSPORT PROGRAM:
Ship 153.0m3    From: S1    To: D2   Traveled: 214.2km
Ship  55.0m3    From: S2    To: D1   Traveled: 132.0km
Ship  97.0m3    From: S2    To: D2   Traveled: 203.7km
Ship  30.0m3    From: S3    To: D1   Traveled:  42.0km
Ship 124.0m3    From: S3    To: D3   Traveled: 458.8km
Ship 138.0m3    From: S4    To: D3   Traveled: 621.0km
Ship 127.0m3    From: S5    To: D3   Traveled: 266.7km
Ship   1.0m3    From: S6    To: D3   Traveled:   6.5km
Ship 128.0m3    From: S6    To: D5   Traveled: 780.8km
Ship  39.0m3    From: S7    To: D4   Traveled: 241.8km
Ship  72.0m3    From: S7    To: D5   Traveled: 410.4km
Ship 110.0m3    From: S8    To: D3   Traveled: 572.0km
Ship 130.0m3    From: S9    To: D1   Traveled: 403.0km
Ship 135.0m3    From:S10    To: D1   Traveled: 432.0km
```

BLOCK 7

Block: AGRICULTURE

What food should be planted so that profit is maximized and the characteristics of the soil, the buyer market and the available resources are respected?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- **Agriculture**
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation
- Livestock
- Rent
- Supplier

Source:

- Book 2
- Page 226

GOAL

A farmer is studying the division of his property in the following productive activities:

A (Let)

To allocate a certain amount of bushel for the sugarcane plantation, to a local power plant, which is in charge of the activity and pays for the rent of the land $300.00 per bushel per year.

P (Livestock)

Use another part for raising cattle. The recovery of pastures requires fertilization (100 kg / bushel) and irrigation (100,000 liters of water per bushel) per year. The estimated profit for this activity is $400.00 per bushel per year.

S (Planting of soybeans)

Use a third part to plant soybeans. This crop requires 200 kg per bushel of fertilizers and 200,000 l of water / acre for irrigation per year. The estimated profit in these activities is $500.00 / bushel in the year.

Resources / Products		Sugar Cane	Cattle	Soy	Available
Water	L	0	100,000	200,000	12,750,000
Fertilizer	kg	0	100	100	14,000
Area	%	1	1	1	100
Profit	$	300.00	400.00	500.00	-

Availability of resources per year: 12,750,000 liters of water, 14,000 kg of fertilizer and 100 bushels of land. How many fields should you allocate to each activity to provide the best return

```
MODEL:
SETS:
  PRODUCT : PROFIT, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE,          AVAILABLE =
  WATER              12750000
  FERTILIZER         14000
  AREA               100;
! Products attributes;
  PRODUCT,           PROFIT =
  CANE               300
  CATTLE             400
  SOY                500;

! Required attributes  CANE      CATTLE     SOY;
  USAGE =              0         100000     200000    ! Water;
                       0         100        100       ! Fertilizer;
                       1         1          1;        ! Area;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',8);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  USAGE(L, Kg, %):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (L, KG, %):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)|PRODUCE(J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J),'-6s'),' Unit profit: $',
        @FORMAT(PROFIT( J),'%6.2f'),' x Area: ',
        @FORMAT(PRODUCE( J),'%5.2f'), 'm2 = Total: $',
        @FORMAT(PROFIT( J) * PRODUCE( J),'%8.2f'),
@NEWLINE( 1));
@CHARTPIE('Agriculture model', 'Area', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
USAGE(L, Kg, %):
                  CANE      CATTLE        SOY
 WATER        0.0000000  100000.00  200000.00
 FERTILIZER   0.0000000  100.00000  100.00000
 AREA         1.0000000  1.0000000  1.0000000

AVAILABLE (L, KG, %):
 WATER        12750000.
 FERTILIZER   14000.000
 AREA         100.00000

PROFIT:
 CANE         300.00000
 CATTLE       400.00000
 SOY          500.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                     42750.000
Infeasibilities:                                     0.0000000

IDEAL PLANNING PROGRAM:
 CANE   Unit profit: $300.00 x Area: 36.25m2 = Total: $10875.00
 SOY    Unit profit: $500.00 x Area: 63.75m2 = Total: $31875.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation

GOAL

A farmer is planning his planting strategy for next year. From information obtained from government agencies, he knows that wheat, rice and corn crops will be the most profitable in the next harvest.

Resources / Products		Wheat	Rice	Corn	Available
Area	m2	400	800	10,000	200,000
Productivity	kg/m2	0.2	0.3	0.4	60,000
Cost p/kg	$	2.16	1.26	0.812	

In the absence of a separate storage site, maximum production is limited to 60 tonnes.

The cultivable area of the site is 200,000 m2, but to meet the demands of its own site it is imperative to plant 400 m2 of wheat, 800 m2 of rice and 10,000 m2 of corn.

It is asked to formulate the problem of optimizing the choice of areas to be cultivated in each different type of crop.

```
MODEL:
SETS:
 PRODUCT : COST, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,              AVAILABLE   =
 AREA                   1200000
 PRODUCTIVITY            60000;

! Products attributes;
 PRODUCT,               COST  =
 WHEAT                  2.16
 RICE                   1.26
 CORN                   0.812;

! Required              WHEAT       RICE        CORN;
 USAGE  =               400         800         10000       ! Area;
                        0.2         0.3         0.4;        ! Productivity;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! Restrictions associated with the site's demand (in unit area m2);
@FOR(PRODUCT(I): PRODUCE(I) >= USAGE(1,I));
      ! Restriction associated with total available area;
      [AVA_AREA] @SUM( PRODUCT( p): PRODUCE( p )) <= 200000;
      ! Restriction associated with storage (in kilograms;
      ! Productivity ratios shall be used per unit area to obtain a final value in kilos;
      [AVA_PROD] @SUM( PRODUCT( p): USAGE( 2, p) * PRODUCE( p )) <= 60000;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE(m2, kg/m2:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (m2, kg/m2):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  COST P/KG:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J): '  ',
      @FORMAT(PRODUCT( J),'-6s'),' Unit cost: $',
      @FORMAT(COST( J),'%4.2f'),'  x  Area:',
      @FORMAT(PRODUCE( J),'%5.0f'), 'm2 = Total: $',
      @FORMAT(COST( J) * PRODUCE( J),'%7.2f'),
@NEWLINE( 1));
@CHARTPIE('Agriculture model', 'Area', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE(m2, kg/m2:
                   WHEAT        RICE        CORN
 AREA            400.00000   800.00000   10000.000
 PRODUCTIVITY  0.20000000  0.30000000  0.40000000

AVAILABLE (m2, kg/m2):
 AREA          1200000.0
 PRODUCTIVITY  60000.000

COST P/KG:
 WHEAT         2.1600000
 RICE          1.2600000
 CORN          0.8120000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                 9992.0000
Infeasibilities:                                 0.0000000

IDEAL PLANNING PROGRAM:
WHEAT  Unit cost: $  2.16  x  Area:  400m2 = Total: $    864.00
RICE   Unit cost: $  1.26  x  Area:  800m2 = Total: $   1008.00
CORN   Unit cost: $  0.81  x  Area:10000m2 = Total: $   8120.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation
- Supplier

GOAL

The government has put 14 hectares of deforested land at the disposal of local farmers.

Resources / Products		Soy	Cotton	Available
Area Available	ha	1	1	14
Men Time by/ha	hr	20	120	1,200
Credit Line by/ha	$	1,200	400	12,000
Profit by/ha	$	100.00	50.00	-

Ensure that such area is used for planting soybeans and cotton. It is estimated that there are 1200 man-hours available during the sowing period; and that it takes 20 man-hours per hectare of soy and 120 per hectare of cotton.

It also offers a maximum credit line of $12,000, divided as follows: $1,200 per hectare of soy and $400 per hectare of cotton.It is desired to organize this area of plantation in order to obtain maximum profit, knowing that the expected profit margins are $100 per hectare of soy and $50 per hectare of cotton.

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
!Resource attribute;
 RESOURCE,          AVAILABLE =
 AREA               14
 MEN_TIME           1200
 CREDIT             12000;

! product attributes;
 PRODUCT,           PROFIT  =
 SOY                100
 COTTON             50;

! Required          SOY       COTTON;
 USAGE      =       1         1              ! Area;
                    20        120            ! Men time by hectare;
                    1200      400;           ! Credit line by hectare;
ENDDATA
SUBMODEL MAX3:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! RESTRICTION OF MAXIMUM HECTAR LIMIT FOR PLANTS;
[AVA_AREA] @SUM( PRODUCT( p): PRODUCE( p )) <= AVAILABLE(1);
! RESTRICTION OF MAN-HOURS BY SOYBEAN HECTARES;
[AVA_HRS]    @SUM(PRODUCT(I): USAGE(2,I)* PRODUCE(I)) <= AVAILABLE(2);
! RESTRICTION OF MAXIMUM CREDIT LIMIT $12 THOUSAND ;
! $1200 PER HECTAR OF SOYA / $400  PER HECTAR OF COTTON;
[AVA_CRE]    @SUM( PRODUCT( p): USAGE( 3, p) * PRODUCE( p )) <= AVAILABLE(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE(m2, kg/m2:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (ha, hr, $):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT BY HECTARE:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J): '  ',
        @FORMAT(PRODUCT( J),'-6s'),' Unit profit: $',
        @FORMAT(PROFIT( J),'%6.2f'),'  x  Area:',
        @FORMAT(PRODUCE( J),'%2.0f'), 'ha  = Total: $',
        @FORMAT(PROFIT( J) * PRODUCE( J),'%6.2f'),
@NEWLINE( 1));
@CHARTPIE('Agriculture model', 'Area', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE(m2, kg/m2:
                 SOY      COTTON
 AREA      1.0000000  1.0000000
 MEN_TIME  20.000000  120.00000
 CREDIT    1200.0000  400.00000

AVAILABLE (ha, hr, $):
 AREA      14.000000
 MEN_TIME  1200.0000
 CREDIT    12000.000

PROFIT BY HECTARE:
 SOY       100.00000
 COTTON    50.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                  1100.0000
Infeasibilities:                                  0.0000000

IDEAL PLANNING PROGRAM:
SOY    Unit profit: $100.00  x  Area: 8ha  = Total: $800.00
COTTON Unit profit: $ 50.00  x  Area: 6ha  = Total: $300.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation
- Supplier

Source:

- Book 1
- Page 52

GOAL

A farmer is planning his planting strategy for next year. From information obtained from government agencies, he knows that wheat, rice and corn crops will be the most profitable in the next harvest.

From experience, you know that the productivity of your land for the crops you want is on the assumptions.

The total availability of area on the farm is 400 hectares to be used, $ 500 thousand in cash and 10 thousand men / day, in addition each hectare of corn generates a profit of $ 600.00 ... as can be observed below:

Resources / Products		Corn	Wheat	Soy	Sugar Cane	Available
Man Day	head	20	30	25	28	10,000
Preparing Earth	$	1,000.00	1,200.00	1,500.00	1,200.00	500,000.00
Area	ha	1	1	1	1	400
Profit by/ha	$	600.00	800.00	900.00	500.00	-

It is asked to formulate the problem of optimizing the choice of areas to be cultivated in each different type of crop.

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE ,       AVAILABLE    =
 MAN_DAY          10000
 PREP_EARTH       500000
 AREA             400;
! Product attributes;
 PRODUCT,         PROFIT       =
 CORN             600
 WHEAT            800
 SOY              900
 CANE             500;

! Required        CORN       WHEAT      SOY        CANE;
 USAGE      =     20         30         25         28          ! Work man men/day;
                  1000       1200       1500       1200        ! Preparing earth;
                  1          1          1          1;          ! Area available;
ENDDATA
SUBMODEL MAX4:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! Manpower 10,000 (Men/day);
[      AVA_MANP] @SUM( PRODUCT( p): USAGE( 1, p) * PRODUCE( p )) <= AVAILABLE(1);
! Cost of preparing the land $ 500 thousand;
       [AVA_PREP]   @SUM(PRODUCT(I): USAGE(2,I)* PRODUCE(I)) <= AVAILABLE(2);
! Area of 400 hectares available;
       [AVA_AREA]  @SUM( PRODUCT( p): PRODUCE( p )) <= AVAILABLE(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE(head, $, ha):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (head, $, ha):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT BY HECTARE:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX4);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
       @FORMAT(PRODUCT( J),'-6s'),'Unit profit:$',
       @FORMAT(PROFIT( J),'%6.2f'),' x Area:',
       @FORMAT(PRODUCE( J),'%3.0f'), 'ha = Total:$',
       @FORMAT(PROFIT( J) * PRODUCE( J),'%7.2f'),
@NEWLINE( 1));
@CHARTPIE('Agriculture model', 'Area', PRODUCE);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE(head, $, ha):
                  CORN       WHEAT         SOY        CANE
 MAN_DAY     20.000000   30.000000   25.000000   28.000000
 PREP_EARTH  1000.0000   1200.0000   1500.0000   1200.0000
 AREA        1.0000000   1.0000000   1.0000000   1.0000000

AVAILABLE (head, $, ha):
 MAN_DAY     10000.000
 PREP_EARTH  500000.00
 AREA        400.00000

PROFIT BY HECTARE:
 CORN        600.00000
 WHEAT       800.00000
 SOY         900.00000
 CANE        500.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                313333.33
Infeasibilities:                                0.0000000

IDEAL PLANNING PROGRAM:
WHEAT Unit profit:$800.00 x Area:167ha = Total:$133333.33
SOY   Unit profit:$900.00 x Area:200ha = Total:$180000.00
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation
- Supplier

Source:

- Book 2
- Page 241

GOAL

A farm has 100 acres of land where it grows watermelon and melon.. Each acre of watermelon requires 150 liters of water per day and 20 kg of fertilizer.

It is estimated that it will take two hours of labor for each acre of watermelon and two and a half hours of labor for each acre of melon.

There is a forecast to sell each watermelon for $3 and each melon for $1. Each acre of watermelon produces 90 units and each acre of melon produces 300 units.

The farmer can pump 18,000 liters of water per day, can buy fertilizers for $10 a bag of 50 kg and can hire temporary labor for $5 an hour.

<table>
<tr><th colspan="3">Resources / Products</th><th>Watermelon</th><th>Melon</th><th>Available</th></tr>
<tr><td rowspan="3">Fertilizer</td><td>Required for acre</td><td>kg</td><td>20</td><td>15</td><td>3,500</td></tr>
<tr><td>Bag w/50 kg</td><td>$</td><td colspan="2">10.00</td><td>700.00</td></tr>
<tr><td>Cost p/acre</td><td>$</td><td>3.00</td><td>2.00</td><td>-</td></tr>
<tr><td rowspan="2">Manpower</td><td>Required for acre</td><td>hr</td><td>2.0</td><td>2.5</td><td>450.0</td></tr>
<tr><td>Cost/hr</td><td>$</td><td>5.00</td><td>5.00</td><td>2,250.00</td></tr>
<tr><td colspan="2">Demand for acre</td><td>un</td><td>90</td><td>300</td><td>-</td></tr>
<tr><td colspan="2">Unity Price</td><td>$</td><td>3.00</td><td>1.00</td><td>-</td></tr>
<tr><td colspan="2">Sale Estimate</td><td>un</td><td>270</td><td>300</td><td>-</td></tr>
</table>

Assuming that the farmer can the whole harvest of watermelon and melon, determine how many acres of each crop the farmer must plant in order to maximize the profit. Water cost were not considered.

```
MODEL:
SETS:
  PRODUCT : PRICE,  CFERT,  CMANP,  SALE,  DEMAND,  PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE,          AVAILABLE    =
  FERT_ACRE          3500
  FERT_COST          700
  MANP_HR            450
  MANP_COST          2250;
! Product attributes;
  PRODUCT,           PRICE,        CFERT,        CMANP,        SALE,        DEMAND =
  WATERMELON         3             3             2             270          90
  MELON              1             2             2.5           300          300;

! Required           WATERMELON                  MELON;
  USAGE =            20                          15                   ! (kg) Fertilizer by acre;
                     10                          10                   ! ( $) Fertilizer bag with 50kg ;
                     2                           2.5                  ! (hr) Manpower by acre;
                     5                           5;                   ! ( $) Manpower PROFIT hour;
ENDDATA
SUBMODEL MAX5:
[OBJ] MAX = SALE(1) * PRODUCE(1) + SALE(2) * PRODUCE(2) - (USAGE(1,1)*CFERT(1)) * PRODUCE(1) -
                     (USAGE(1,2)*CFERT(2)) * PRODUCE(2) - (CMANP(1)*5) * PRODUCE(1) - (CMANP(2)*5) * PRODUCE(2);
! Area of 100 hectares available;
[AREA] @SUM( PRODUCT( p): PRODUCE( p )) <= 100;
! Demand;
[DEM] DEMAND(1)* PRODUCE(1) + DEMAND(2) * PRODUCE(2) <=  (DEMAND(1) + DEMAND(2)) * 100;
! Manpower consumption;
[MAN] USAGE(3,1) * PRODUCE(1) + USAGE(3,2) * PRODUCE(2) <= AVAILABLE(3);
! Consumption of fertilizers for 100 acres;
[FER] USAGE(1,1) * PRODUCE(1) + USAGE(1,2) * PRODUCE(2) <= AVAILABLE(1);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',8);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE(Kg, $, hr, $):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1),   "  AVAILABLE (kg, $, hr, $):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND per Acre:", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  UNIT PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SALE ESTIMATED:", @NEWLINE( 1));
@TABLE(SALE);
@WRITE(" ", @NEWLINE( 1), "  COST OF FERTILIZER p/acre:", @NEWLINE( 1));
@TABLE(CFERT);
@WRITE(" ", @NEWLINE( 1), "  COST OF LABOR p/acre:", @NEWLINE( 1));
@TABLE(CMANP);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute Sub-model;
@SOLVE(MAX5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE( J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J),'-9s'),@NEWLINE(1),'   Price:$',
        @FORMAT(SALE( J),'%6.2f'),' x Planted:',
        @FORMAT(PRODUCE(J),'%3.0f'), ' Acres = Revenue:$',
        @FORMAT(PRODUCE( J)* PRICE(J) * DEMAND(J),'%7.2f'), @NEWLINE(1),42*' ','Cost:$',
        @FORMAT(CFERT(J)*USAGE(1,J)*PRODUCE(J) + CMANP(J)*USAGE(4,J)*PRODUCE(J),'%8.2f'),
        @NEWLINE(1),40*' ','Profit:$',
        @FORMAT((PRODUCE( J)* PRICE(J)* DEMAND(J)) - (CFERT(J)*USAGE(1,J)*PRODUCE(J) +
        CMANP(J)*USAGE(4,J)*PRODUCE(J)),'%8.2f'),
@NEWLINE(1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE(Kg,$,hr,$):
            WATERMELON       MELON
 FERT_ACRE   20.000000   15.000000
 FERT_COST   10.000000   10.000000
 MANP_HR     2.0000000   2.5000000
 MANP_COST   5.0000000   5.0000000

AVAILABLE (kg,$,hr,$):
 FERT_ACRE  3500.0000
 FERT_COST  700.00000
 MANP_HR    450.00000
 MANP_COST  2250.0000

DEMAND p/acre:
 WATERMELON  90.000000
 MELON       300.00000

UNIT PRICE:
 WATERMELON  3.0000000
 MELON       1.0000000

SALE ESTIMATED:
 WATERMELON  270.00000
 MELON       300.00000

COST OF FERTILIZER p/acre:
 WATERMELON  3.0000000
 MELON       2.0000000

COST OF LABOR p/acre:
 WATERMELON  2.0000000
 MELON       2.5000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                             25750.000
Infeasibilities:                             0.0000000

IDEAL PLANNING PROGRAM:
MELON
 Price:$300.00 x Planted:100 Acres = Revenue:$30000.00
                                        Cost:$ 4250.00
                                      Profit:$25750.00
```

6

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Farm
- Plantation
- Supplier

GOAL

A reforestation company has plantation areas in 4 municipalities. The company considers the use of 4 tree species: Pines, Oak, Walnut and Araucaria. The data for elaboration of the model are below:

Expected Annual Income					
Resources / Products		Pine	Oak	Walnut	Araucaria
City A	$/he	16.00	12.00	20.00	18.00
City B	$/he	14.00	13.00	24.00	20.00
City C	$/he	17.00	10.00	28.00	20.00
City D	$/he	12.00	11.00	18.00	17.00

Expected Annual Production						
Resources / Products		Pine	Oak	Walnut	Araucaria	Available
City A	m3/he	17	14	10	9	1,500
City B	m3/he	15	16	12	11	1,700
City C	m3/he	13	12	14	8	900
City D	m3/he	10	11	8	6	600
Minimum Production	m3/he	22,500	9,000	28,800	3,500	-

Formulate the problem to designate the areas of planting by municipality in order to maximize the income.

```
MODEL:
SETS:
 PRODUCT : PROD_MIN;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT): INCOME, PRODUCTION, PRODUCE;
ENDSETS
DATA:
! Resources (Available Area (m3/Hectare) ;
 RESOURCE,          AVAILABLE    =
 CITY_A             1500
 CITY_B             1700
 CITY_C             900
 CITY_D             600;
 ! Products attributes;
 PRODUCT,           PROD_MIN  =
 PINUS              22500
 OAK                9000
 WALNUT             28800
 ARAUCARIA          3500;
! Required          PINUS OAK   WALNUT      ARAUCARIA;
 INCOME      =      16     12    20          18          ! Expected annual income City A;
                    14     13    24          20          ! Expected annual income City B;
                    17     10    28          20          ! Expected annual income City C;
                    12     11    18          17;         ! Expected annual income City D;
 PRODUCTION =       17     14    10          9           ! Expected annual production City A;
                     15    16    12          11          ! Expected annual production City B;
                    13     12    14          8           ! Expected annual production City C;
                    10     11    8           6;          ! Expected annual production City D;
ENDDATA
SUBMODEL MAX6:
MAX = @SUM(PRODUCT(J): INCOME(1,J) * PRODUCE(1,J)) + @SUM(PRODUCT(J): INCOME(2,J) * PRODUCE(2,J)) +
        @SUM(PRODUCT(J): INCOME(3,J) * PRODUCE(3,J)) + @SUM(PRODUCT(J): INCOME(4,J) * PRODUCE(4,J)) ;
! Available Area (m3/Hectare);
[AVA1] @SUM(PRODUCT(J): PRODUCE(1,J)) = AVAILABLE(1);
[AVA2] @SUM(PRODUCT(J): PRODUCE(2,J)) = AVAILABLE(2);
[AVA3] @SUM(PRODUCT(J): PRODUCE(3,J)) = AVAILABLE(3);
[AVA4] @SUM(PRODUCT(J): PRODUCE(4,J)) = AVAILABLE(4);
! Minimal Production;
[MP1] @SUM( RESOURCE(I): PRODUCTION(I,1) * PRODUCE(I,1)) >= PROD_MIN(1);
[MP2] @SUM( RESOURCE(I): PRODUCTION(I,2) * PRODUCE(I,2)) >= PROD_MIN(2);
[MP3] @SUM( RESOURCE(I): PRODUCTION(I,3) * PRODUCE(I,3)) >= PROD_MIN(3);
[MP4] @SUM( RESOURCE(I): PRODUCTION(I,4) * PRODUCE(I,4)) >= PROD_MIN(4);
ENDSUBMODEL
CALC:
@SET('TERSEO',1);           ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);           ! Post status windows, 1 Yes, 0 No;
@SET('PRECIS',8);           ! Precision in digits for standard solution reports;
@WRITE("  DATA:", @NEWLINE( 1), "  EXPECTED ANNUAL INCOME ($/he):", @NEWLINE( 1));
@TABLE(INCOME);
@WRITE(" ", @NEWLINE( 1), "  EXPECTED ANNUAL PRODUCTION (m3/he):", @NEWLINE( 1));
@TABLE(PRODUCTION);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (m3/he):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM PRODUCTION (m3/he):", @NEWLINE( 1));
@TABLE(PROD_MIN);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX6)
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP(i,j) : '  ',
        @FORMAT(RESOURCE( I),'-6s'),', Production: ',
        @FORMAT(PRODUCE(I,J),'%8.3f'), 'ha of ',
        @FORMAT(PRODUCT( J), '-9s'),' x Income: $',
        @FORMAT(INCOME(I,J),'%4.2f'),'  =  Revenue: $',
        @FORMAT(PRODUCE(I,J) * INCOME( I,J),'%8.2f'),
@NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
EXPECTED ANNUAL INCOME ($/he):
             PINUS        OAK     WALNUT   ARAUCARIA
 CITY_A  16.000000  12.000000  20.000000  18.000000
 CITY_B  14.000000  13.000000  24.000000  20.000000
 CITY_C  17.000000  10.000000  28.000000  20.000000
 CITY_D  12.000000  11.000000  18.000000  17.000000

EXPECTED ANNUAL PRODUCTION (m3/he):
             PINUS        OAK     WALNUT   ARAUCARIA
 CITY_A  17.000000  14.000000  10.000000  9.0000000
 CITY_B  15.000000  16.000000  12.000000  11.000000
 CITY_C  13.000000  12.000000  14.000000  8.0000000
 CITY_D  10.000000  11.000000  8.0000000  6.0000000

AVAILABLE (m3/he):
 CITY_A  1500.0000
 CITY_B  1700.0000
 CITY_C  900.00000
 CITY_D  600.00000

MINIMUM PRODUCTION (m3/he):
 PINUS       22500.000
 OAK         9000.0000
 WALNUT      28800.000
 ARAUCARIA   3500.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                     94827.882
Infeasibilities:                                                     0.0000000

IDEAL PLANNING PROGRAM:
CITY_A, Production: 1323.529 m3/he of PINUS     x Income: $16.00 = Revenue: $21176.47
CITY_A, Production:  176.471 m3/he of OAK       x Income: $12.00 = Revenue: $ 2117.65
CITY_A, Production:    0.000 m3/he of WALNUT    x Income: $20.00 = Revenue: $    0.00
CITY_A, Production:    0.000 m3/he of ARAUCARIA x Income: $18.00 = Revenue: $    0.00
CITY_B, Production:    0.000 m3/he of PINUS     x Income: $14.00 = Revenue: $    0.00
CITY_B, Production:  170.941 m3/he of OAK       x Income: $13.00 = Revenue: $ 2222.24
CITY_B, Production: 1350.000 m3/he of WALNUT    x Income: $24.00 = Revenue: $32400.00
CITY_B, Production:  179.059 m3/he of ARAUCARIA x Income: $20.00 = Revenue: $ 3581.18
CITY_C, Production:    0.000 m3/he of PINUS     x Income: $17.00 = Revenue: $    0.00
CITY_C, Production:    0.000 m3/he of OAK       x Income: $10.00 = Revenue: $    0.00
CITY_C, Production:  900.000 m3/he of WALNUT    x Income: $28.00 = Revenue: $25200.00
CITY_C, Production:    0.000 m3/he of ARAUCARIA x Income: $20.00 = Revenue: $    0.00
CITY_D, Production:    0.000 m3/he of PINUS     x Income: $12.00 = Revenue: $    0.00
CITY_D, Production:  344.941 m3/he of OAK       x Income: $11.00 = Revenue: $ 3794.35
CITY_D, Production:    0.000 m3/he of WALNUT    x Income: $18.00 = Revenue: $    0.00
CITY_D, Production:  255.059 m3/he of ARAUCARIA x Income: $17.00 = Revenue: $ 4336.00
```

7

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation
- Supplier

GOAL

A farmer has a land area of 5 km2 to be sown with Wheat and Barley. It has a limited amount of Fertilizer and Insecticide allowed to be used. Both Wheat and Barley require different amounts per unit area planted.

Resources / Products		Barley	Wheat	Available
Fertilizer	kg	30	20	50
Insecticide	kg	15	10	22
Area	km2	3	2	5
Price p/km2	$	16,000.00	12,000.00	-

Prices expected to be sold should be $16,000 and $12,000, respectively for Wheat and Barley per Km2. Develop a Linear Programming model in order to obtain the best recipe for this combination of planting.

```
MODEL:
SETS:
 HEADER1 / PROD, LIMIT, VALUE /:;
 PRODUCT : PRICE, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
 PXR( RESOURCE, HEADER1) : SLASUR;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE ,              AVAILABLE    =
 FERTILIZER              50
 INSECTICIDE             22
 AREA                    5;
! Product attributes;
 PRODUCT,                PRICE =
 WHEAT                   16000
 BARLEY                  12000;

! Required               WHEAT        BARLEY;
 USAGE       =           30           20           ! Fertilizer Limit;
                         15           10           ! Insecticide limit;
                         3            2;           ! Area available in km2;
ENDDATA
SUBMODEL MAX7:
[OBJ] MAX =  @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! Fertilizer Availability Constraints;
[FERT] @SUM( PRODUCT( p): USAGE( 1, p) * PRODUCE( p )) <= AVAILABLE(1);
! Insecticide Availability Constraints;
[INSEC]      @SUM(PRODUCT(I): USAGE(2,I)* PRODUCE(I)) <= AVAILABLE(2);
! Area Availability Constraints (5km2);
[AVA_AREA]   @SUM( PRODUCT( p): PRODUCE( p )) <= AVAILABLE(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',8);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE(Kg, Kg, Km2):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (head, $, ha):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE BY KM2:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX7);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J) | PRODUCE( J) #GT# 0: '  ',
      @FORMAT(PRODUCT( J),'-7s'),' Price:$',
      @FORMAT(PRICE( J),'%6.2f'),' x ',
      @FORMAT(PRODUCE( J),'%4.2f'), ' Km2 = Revenue:$',
      @FORMAT(PRICE( J) * PRODUCE( J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE(Kg, Kg, Km2):
                  WHEAT     BARLEY
 FERTILIZER   30.000000  20.000000
 INSECTICIDE  15.000000  10.000000
 AREA         3.0000000  2.0000000

AVAILABLE (head, $, ha):
 FERTILIZER   50.000000
 INSECTICIDE  22.000000
 AREA         5.0000000

PRICE BY KM2:
 WHEAT   16000.000
 BARLEY  12000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              26400.000
Infeasibilities:                              0.0000000

IDEAL PLANNING PROGRAM:
BARLEY  Price:$12000.00 x 2.20 Km2 = Revenue:$26400.00
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Farm
- Plantation

Source:

- Book 1
- Page 55

GOAL

A farmer has 200 hectares of arable land for corn and / or soy. The data are as follows:

Resources / Products		Corn	Soy	Available
Area	he	1	1	200
Preparation	$	500.00	700.00	200,000.00
Labor	men/day	15	18	20,000
Profit p/he	$	900.00	1,300.00	-

What should be the allocation of land to the various types of crop in order to maximize profits?

```
MODEL:
SETS:
  PRODUCT : PROFIT, PRODUCE;
  RESOURCE: MINREQ;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attribute (hectares, $, Men/Day;
  RESOURCE,                    MINREQ =
  AREA                         200
  PREPARATION                  200000
  LABOR                        20000;
! Product attributes;
  PRODUCT,                     PROFIT  =
  CORN                         900
  SOY                          1300;

! Required                     CORN         SOY;
  USAGE       =                1            1            ! AREA;
                               500          700          ! PREPARATION;
                               15           18;          ! LABOR;
ENDDATA
SUBMODEL MAX8:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The Area constraints;
[AVA_AREA] @SUM( PRODUCT( p): PRODUCE( p )) <= MINREQ(1);
! The Preparation Constraints;
[AVA_PRE]    @SUM(PRODUCT(I): USAGE(2,I)* PRODUCE(I)) <= MINREQ(2);
! The Labor constraints;
[AVA_LAB]   @SUM( PRODUCT( p): USAGE( 3, p) * PRODUCE( p )) <= MINREQ(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',8);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE :", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  PROFIT BY HECTARE:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX8);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J): '   ',
        @FORMAT(PRODUCT( J),'-5s'),'   Profit: $',
        @FORMAT(PROFIT( J),'%7.2f'),' x ',
        @FORMAT(PRODUCE( J),'%3.0f'), ' ha = Total: $',
        @FORMAT(PROFIT( J) * PRODUCE( J),'%9.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX8);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE:
                   CORN        SOY
 AREA          1.0000000  1.0000000
 PREPARATION   500.00000  700.00000
 LABOR         15.000000  18.000000

AVAILABLE:
 AREA          200.00000
 PREPARATION   200000.00
 LABOR         20000.000

PROFIT BY HECTARE:
 CORN          900.00000
 SOY           1300.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                             260000.00
Infeasibilities:                             0.0000000

IDEAL PLANNING PROGRAM:
 CORN    Profit: $ 900.00 x   0 ha = Total: $     0.00
 SOY     Profit: $1300.00 x 200 ha = Total: $260000.00
```

BLOCK 8

Block: CONSTRUCTION

How to know the best number of apartment houses according to the resources available for Manpower?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- **Construction**
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- House
- Apartment

GOAL

What is the ideal amount of popular homes and apartments that the builder must build to get the maximum profit considering the information below.

Resources / Products		House	Apartments	Available
Bricklayer	head	3	2	30
Servant	head	4	3	70
Woodwork	head	3	2	30
Plumber	head	2	1	20
Electrician	head	1	1	20
Painter	head	3	2	20
Gardening	head	1	0	20
Profit p/unit	$	6,120.00	4,100.00	-

For a strategic reason of the constructor, at least three houses must be made.

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: AVAILABLE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        AVAILABLE    =
 BRICKLAYER       30
 SERVANT          70
 WOODWORK         30
 PLUMBER          20
 ELECTRICIAN      20
 PAINTER          20
 GARDENING        20;
! Products attributes;
 PRODUCT,         PROFIT =
 HOUSE            6120
 APTO             4100;
! Required        HOUSE       APTO;
 USAGE  =         3           2           ! BRICKLAYER;
                  4           3           ! SERVANT;
                  3           2           ! WOODWORK;
                  2           1           ! PLUMBER;
                  1           1           ! ELECTRICIAN;
                  3           2           ! PAINTER;
                  1           0;          ! GARDENING;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
! Minimal Construction of Houses;
PRODUCE(1) = 3;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (head)):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Head)):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE(J) #GT# 0: '  Build:',
        @FORMAT(PRODUCE( J),'%2.0f'),
        @FORMAT(PRODUCT( J),'-6s'),'x Unit profit:$',
        @FORMAT(PROFIT( J), '%6.2f'), ' = Total:$',
        @FORMAT(PROFIT( J) * produce( J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (head):
                 HOUSE      APTO
 BRICKLAYER   3.000000  2.000000
 SERVANT      4.000000  3.000000
 WOODWORK     3.000000  2.000000
 PLUMBER      2.000000  1.000000
 ELECTRICIAN  1.000000  1.000000
 PAINTER      3.000000  2.000000
 GARDENING    1.000000  0.000000

AVAILABLE (Head)):
 BRICKLAYER   30.00000
 SERVANT      70.00000
 WOODWORK     30.00000
 PLUMBER      20.00000
 ELECTRICIAN  20.00000
 PAINTER      20.00000
 GARDENING    20.00000

PROFIT:
 HOUSE  6120.000
 APTO   4100.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                               40910.00
Infeasibilities:                               0.000000

IDEAL PLANNING PROGRAM:
Build: 3 HOUSE x Unit profit:$6120.00 = Total:$18360.00
Build: 6 APTO  x Unit profit:$4100.00 = Total:$22550.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- House
- Apartment

Source:

- Book 11

GOAL

It is intended to optimize profits with constructions of standard buildings of area of 80 m2,100 m2 and 120 m2 in the same allotment.

You must determine the quantities of standard buildings so that the profit in the enterprise is the maximum.

Resources / Products		Apt 80 m2	Apt100 m2	Apt 120 m2	Available
Man Hours	hr	3,000	2,000	4,000	300,000
Profit	$	6,000.00	4,200.00	7,500.00	-

The profit per unit of standard building of 80 m2, 100 m2 and 120 m2 is, respectively, $6,000.00, $4,200.00 and $7,500.00.

Being the lot with 120 lots and the standard constructions of 80 m2, 100 m2 and 120 m2 consume 3,000, 2,000 and 4,000 man-hours per building and the number of men available now is 300,000 hours.

And that, for the service of the sales department, there are 15 units of 80 m2 already sold at the launch of the allotment.

```
MODEL:
SETS:
 RESOURCE: AVAILABLE, LOTS;
 PRODUCT : PROFIT, PRODUCE;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Members attributes;
 RESOURCE,          AVAILABLE,      LOTS  =
 MAN_HOURS          300000          120;
! Products attributes;
 PRODUCT,           PROFIT =
 APTO_80            6000
 APTO_100           4200
 APTO_120           7500;

! Required          APTO_80    APTO_100    APTO_120;
 USAGE  =           3000       2000        4000;       ! Man-Hours required;

ENDDATA
SUBMODEL MAX2:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
      [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
! The lots constraints;
@FOR( RESOURCE( r):
      [LOT] @SUM( PRODUCT( p): PRODUCE( p )) = LOTS( r););
! Number of apartments already sold ;
[MIN_APTO_80] PRODUCE( 1 ) >= 15;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  REQUIRED: ", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  LOTS:", @NEWLINE( 1));
@TABLE(LOTS);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE(J) #GT# 0: '  Build: ',
      @FORMAT(PRODUCE( J),'%5.0f'), ' ',
      @FORMAT(PRODUCT( J),'-8s'),' x Unit profit: $',
      @FORMAT(PROFIT( J), '%7.2f'), ' = Total: $',
      @FORMAT(PROFIT( J) * produce( J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
REQUIRED:
              APTO_80  APTO_100  APTO_120
 MAN_HOURS  3000.000  2000.000  4000.000

AVAILABLE:
 MAN_HOURS  300000.0

LOTS:
 MAN_HOURS  120.0000

PROFIT:
 APTO_80    6000.000
 APTO_100   4200.000
 APTO_120   7500.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                       612000.0
Infeasibilities:                                       0.000000

IDEAL PLANNING PROGRAM:
Build:    60 APTO_80  x Unit profit: $6000.00 = Total: $360000.00
Build:    60 APTO_100 x Unit profit: $4200.00 = Total: $252000.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- House
- Apartment

Source:

- Book 7

GOAL

What is the ideal amount of popular homes and apartments that the builder must build to get the maximum profit considering the information below.

Resources / Products		House	Apartments	Available
Bricklayer	head	4	8	30
Servant	head	2	3	70
Woodwork	head	1	3	20
Profit p/unit	$	3,000.00	5,000.00	-

```
MODEL:
SETS:
  PRODUCT : PROFIT, PRODUCE;
  RESOURCE: AVAILABLE;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE,        AVAILABLE    =
  BRICKLAYER       30
  SERVANT          70
  WOODWORK         20;
! Products attributes;
  PRODUCT,         PROFIT =
  HOUSE            3000
  APTO             5000;
! Required          HOUSE         APTO;
  USAGE  =          2             3             ! BRICKLAYER;
                    4             8             ! SERVANT;
                    1             3;            ! WOODWORK;
ENDDATA
SUBMODEL MAX3:
[OBJ] MAX = @SUM( PRODUCT( p): PROFIT( p) * PRODUCE( p));
! The available constraints;
@FOR( RESOURCE( r):
        [AVA] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (head)):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Head)):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE(J) #GT# 0: '  Build:',
        @FORMAT(PRODUCE( J),'%2.0f'),
        @FORMAT(PRODUCT( J),'-6s'),'x Unit profit:$',
        @FORMAT(PROFIT( J), '%6.2f'), ' = Total:$',
        @FORMAT(PROFIT( J) * produce( J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (head)):
                HOUSE      APTO
 BRICKLAYER  2.000000  3.000000
 SERVANT     4.000000  8.000000
 WOODWORK    1.000000  3.000000

AVAILABLE (Head)):
 BRICKLAYER  30.00000
 SERVANT     70.00000
 WOODWORK    20.00000

PROFIT:
 HOUSE  3000.000
 APTO   5000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                     46666.67
Infeasibilities:                                     0.000000

IDEAL PLANNING PROGRAM:
Build: 10.00 HOUSE  x Unit profit: $3000.00 = Total: $30000.00
Build:  3.33 APTO   x Unit profit: $5000.00 = Total: $16666.67
```

BLOCK 9

Block: **REFINERY**

What should be the oil mixture being sent to a cracking tower to produce its derivatives (gasoline, oil, etc.) at a minimal cost? Do the oils come from different sources and have different compositions?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- **Refinery**
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Petroleum
- Refine
- Naphtha
- Oil

Source:

- Book 3
- Chapter 2.3.7

GOAL

A small refinery produces naphtha and gasoline, according to productivity data, gasoline, naphtha and crude oil, as can be seen below.

Resources / Products		Gasoline	Naphtha	Capacity
Petroleum	m3/gal	0.4	0.5	300
Refine	Ut/gal	1	2	900
Demand	gal	-	375	-
Stock	gal	600	-	-
Production Scale	gal	150	150	-
Profit	$	3.00	5.00	-

The time needed to refine one gallon for each different type of fuel, and also defines in the planning horizon, a minimum production scale for the operation of the refinery, the maximum capacity of gasoline stock and the maximum demand for the market of naphtha .

Schedule the operation of the refinery in order to maximize profit from the sale of manufactured fuels.

```
MODEL:
SETS:
PRODUCT:DEMAND, STOCK, SCALE, PRODUCE, PROFIT ;
RESOURCE: CAPACITY;
ROUTES( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Products attributes;
PRODUCT,          DEMAND,     STOCK,      SCALE,      PROFIT      =
GASOLINE          0           600         150         3
NAPHTHA           375         0           150         5;
! Resource attributes;
RESOURCE,         CAPACITY  =
PETROLEUM         300
REFINE            900;

! Required        GASOLINE    NAPHTHA;
USAGE       =     0.4         0.5         ! PETROLEUM;
                  1           2;          ! REFINE;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX = @SUM( PRODUCT( J): PROFIT(J) * PRODUCE( J));
! The capacity constraints;
@FOR(PRODUCT(I):
        [CAP] @SUM( RESOURCE( J): USAGE( I, J) * PRODUCE(J)) <= CAPACITY( I));
! The production scale constraints;
@FOR(PRODUCT(I): [SCA] PRODUCE(I) >= SCALE( I));
! The demand constraints;
@FOR(PRODUCT(I) | DEMAND(I) #GT# 0:[DEM] PRODUCE(I) <= DEMAND(I));
! The Stock constraints;
@FOR(PRODUCT(I) | STOCK(I)  #GT# 0:[STK] PRODUCE(I) <= STOCK(I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  USAGE (m3/gal, Ut/gal):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  CAPACITY (m3/gal, Ut/gal):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (gal):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  STOCK (gal):", @NEWLINE( 1));
@TABLE(STOCK);
@WRITE(" ", @NEWLINE( 1), "  PRODUCTION SCALE (gal):", @NEWLINE( 1));
@TABLE(SCALE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J): '  ',
        @FORMAT(PRODUCT( J),'-8s'),' Production:',
        @FORMAT(PRODUCE( J), '%3.0f'),'gal x Unit profit: $',
        @FORMAT(PROFIT( J),'%4.2f'), ' = Revenue: $',
        @FORMAT(PROFIT( J)  * PRODUCE( J),'%7.2f'),
@NEWLINE(1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
USAGE (m3/gal, Ut/gal):
             GASOLINE    NAPHTHA
 PETROLEUM  0.4000000  0.5000000
 REFINE      1.000000   2.000000

CAPACITY (m3/gal, Ut/gal):
 PETROLEUM  300.0000
 REFINE     900.0000

DEMAND (gal):
 GASOLINE   0.000000
 NAPHTHA    375.0000

STOCK (gal):
 GASOLINE   600.0000
 NAPHTHA    0.000000

PRODUCTION SCALE (gal):
 GASOLINE   150.0000
 NAPHTHA    150.0000

PROFIT:
 GASOLINE   3.000000
 NAPHTHA    5.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
SOLUTION:
Global optimal solution found.
Objective value:                                           2500.000
Infeasibilities:                                           0.000000

 IDEAL PLANNING PROGRAM:
 GASOLINE Production:500gal x Unit profit: $3.00 = Revenue: $1500.00
 NAPHTHA  Production:200gal x Unit profit: $5.00 = Revenue: $1000.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Commodity
- Gasoline
- Blend
- Product Mix

Source:

- Book 5
- Page 238

GOAL

One additional aspect of blending problem formulation will be illustrated with an example in which the batch size is a decision variable. In the previous example, the batch size was specified.

In the following example, the amount of product to be blended depends upon how cheaply the product can be blended.

Thus, it appears the blending decision and the batch size decision must be made simultaneously.

This example is suggestive of gasoline blending problems faced in a petroleum refinery. We wish to blend gasoline from three ingredients: butane, heavy naphtha, and catalytic reformats.

Four characteristics of the resultant gasoline and its inputs are important: cost, octane number, vapor pressure, and volatility. These characteristics are summarized in the following table:

	Commodity				
Feature	**Butane (BUT)**	**Catalytic Reformate (CAT)**	**Heavy Naphtha (NAP)**	**Regular Gasoline (REG)**	**Premium Gasoline (PRM)**
Cost/Unit	7.3	18.2	12.5	-18.4	-22
Octane	120.0	100.0	74.0	$89 \le oct \le 110$	$94 \le oct \le 110$
Vapor Pressure	60.0	2.6	4.1	$8 \le vp \le 11$	$8 \le vp \le 11$
Volatility	105.0	3.0	12.0	$17 \le vo \le 25$	$17 \le vo \le 25$
Availability	1000.0	4000.0	5000.0	$4000 \le sell \le 8000$	$2000 \le sell \le 6000$

The cost per unit for REG and PRM are listed as negative, meaning we can sell them. That is, a negative cost is a revenue.

The octane rating is a measure of the gasoline's resistance to "knocking" or "pinging". Vapor pressure and volatility are closely related.

Vapor pressure is a measure of susceptibility to stalling, particularly on an unusually warm spring day. Volatility is a measure of how easily the engine starts in cold weather.

From the table, we see in this planning period, for example, there are only 1,000 units of butane available.

The profit contribution of regular gasoline is $18.40 per unit exclusive of the cost of its ingredients.

A slight simplification assumed in this example is that the interaction between ingredients is linear.

For example, if a "fifty/fifty" mixture of BUT and CAT is made, then its octane will be $0.5 \times 120 + 0.5 \times 100 = 110$ and its volatility will be $0.5 \times 105 + 0.5 \times 3 = 54$.

In reality, this linearity is violated slightly, especially with regard to octane rating.

Formulation

The quality constraints require a bit of thought. The fractions of a batch of REG gasoline consisting of Butane, Catalytic Reformat, and Heavy Naphtha are BUT/REG, CAT/REG, and NAP/REG, respectively. Thus, if the god of linearity smiles upon us, the octane constraint of the blend for REG should be the expression:

(BUT/REG) × 120 + (CAT/REG) × 100 + (NAP/REG) × 74 ≥ 89.

Your expression, however, may be a frown because a ratio of variables like BUT/REG is definitely not linear. Multiplying through by REG, however, produces the linear constraint: 120 BUT + 100 CAT + 74 NAP ≥ 89 REG or in standard form:

120 BUT + 100 CAT + 74 NAP - 89 REG ≥ 0.

Representing Two-sided Constraints

All the quality requirements are two sided. That is, they have both an upper limit and a lower limit. The upper limit constraint on octane is clearly:

120 BUT + 100 CAT + 74 NAP - 110 REG ≤ 0.

We can write it in equality form by adding an explicit slack:

120 BUT + 100 CAT + 74 NAP - 110 REG + SOCT = 0.

When SOCT = 0, the upper limit is binding. You can verify that, when SOCT = 110 REG – 89 REG = 21 REG, the lower limit is binding.

Thus, a compact way of writing both the upper and lower limits is with the two constraints:

1) 120BUT+100CAT+74NAP-110REG+SOCT=0,

2) 2) SOCT ≤ 21 REG.

Notice, even though there may be many ingredients, the second constraint involves only two variables. This is a compact way of representing two-sided constraints. Similar arguments can be used to develop the vapor and volatility constraints. Finally, a constraint must be appended, which states the whole equals the sum of its raw material parts, specifically:

REG = BUT + NAP + CAT.

The solution suggests that Premium is the more profitable product, so we sell the minimum amount of Regular required and then sell as much Premium as scarce resources, BUT and CAT, allow. LP blending models have been a standard operating tool in refineries for years.

Recently, there have been some instances where these LP models have been replaced by more sophisticated nonlinear models, which more accurately approximate the nonlinearities in the blending process. See Rigby, Lasdon, and Waren (1995), for a discussion of how Texaco does it.

For example, volatility may be represented by a logarithmic expression and octane may be represented with a polynomial like $a_1*x + a_2*x^2 + a_3*x^3 + a_4*x^4$, see Rardin(1998).

There is a variety of complications as gasoline blending models are made more detailed. For example, in high quality gasoline, the vendor may want the octane to be constant across volatility ranges in the ingredients.

The reason is, if you “floor” the accelerator on a non-fuel injected automobile, a shot of raw gas is squirted into the intake.

The highly volatile components of the blend will reach the combustion chamber first. If these components have low octane, you will have knocking, even though the “average” octane rating of the gasoline is high.

This may be more important in a station selling gas for city driving than in a station on a cross country highway in Kansas where most driving is at a constant speed.

```
 MODEL:
! General Blending Model(BLEND) in LINGO;
SETS:
!Each raw material has availability & cost/unit;
RM/ BUT, CAT, NAP/: A, C;
! Each f. g. has min & max sellable, profit contr./unit and batch size to be determined;
FG/ REG, PRM/: D, E, P, B;
! There are a set of quality measures;
QM/ OCT, VAP, VOL/;
!Each RM & QM combo has a quality level;
RQ( RM, QM): Q;
!For each combo QM, FG there are upper & lower limits on quality, slack on quality to be determined;
QF( QM, FG): U, L, S;
!Each combination of RM and FG has an amount used, to be determined;
RF( RM, FG): X;
ENDSETS

DATA:
! RM    BUT     CAT     NAP;
A =     1000,   4000,   5000;   ! Raw material availabilities;
C =     7.3,    18.2,   12.5;   ! R. M. costs;
! QM    OCT     VAP     VOL;
Q =     120,    60,     105,    ! Quality parameters...;
        100,    2.6,    3,      ! R. M. by quality;
        74,     4.1,    12;
! FG    REG     PRM;
D =     4000,   2000;           ! Min needed of each F.G.;
E =     8000,   6000;           ! Max sellable of each F.G;
P =     18.4,   22;             ! Selling price of each F.G.;
U =     110,    110,            ! Upper limits on quality;
         11,    11,             ! Quality by F.G.;
         25,    25;
! Lower limits on quality...;
! QF    QM      FG;
L=      89,     94
        8,      8,
        17,     17;
ENDDATA
SUBMODEL MAX5:
! For each raw material, the availabilities;
@FOR( RM( I):
     [RMLIM] @SUM( FG( K): X( I, K)) < A( I););
@FOR( FG( K):
    !For each finished good, compute batch size;
    [BDEF] B( K) = @SUM( RM( I): X( I, K));
    ! Batch size limits;
    [BLO] B( K) > D( K);
    [BHI] B( K) < E( K);
    ! Quality restrictions for each quality;
    @FOR( QM( J):
       [QUP]@SUM( RM(I): Q(I, J) * X(I, K)) + S( J,K) = U( J, K) * B( K);
       [QDN] S(J, K) < (U(J, K) - L(J, K)) * B(K);
    );
);
!We want to maximize profit contribution;
[PROFIT] MAX = @SUM( FG: P * B) - @SUM( RM( I): C( I) * @SUM( FG( K): X( I, K)));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
!Precision in digits for standard solution reports;
@SET('PRECIS',6);
! Set page width;
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RAW MATERIAL:", @NEWLINE( 1));
@TABLE(A);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(C);
@WRITE(" ", @NEWLINE( 1), "  QUALITY PARAMETERS :", @NEWLINE( 1));
@TABLE(Q);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM NEED OF EACH F.G. :", @NEWLINE( 1));
@TABLE(D);
@WRITE(" ", @NEWLINE( 1), "  MAXIMUM SELLABLE OF EACH F.G. :", @NEWLINE( 1));
@TABLE(E);
@WRITE(" ", @NEWLINE( 1), "  SELLING PRICE OF EACH F.G. :", @NEWLINE( 1));
@TABLE(P);
@WRITE(" ", @NEWLINE( 1), "  UPPER LIMITS ON QUALITY BY F.G. :", @NEWLINE( 1));
@TABLE(U);
@WRITE(" ", @NEWLINE( 1), "  LOWER LIMITS ON QUALITY BY F.G. :", @NEWLINE( 1));
@TABLE(L);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE( MAX5 );
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
! Cost;
@WRITE("  TOTAL COST:", @NEWLINE( 1));
@WRITEFOR( RF(I,J): '  ',RM(J),'     Used: ',
        @FORMAT(X(I, J),'%6.1f'), ' x Cost: $',
        @FORMAT(C(I),'%6.2f'), ' = Total:$',
        @FORMAT(X(I,J) * C(I),'%10.2f'),
@NEWLINE( 1));
@WRITE('  TOTAL (C):',35*' ','$ ',
        @FORMAT(@SUM(RF(I,J): X(I,J) * C(I)),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
! Price;
@WRITE("  TOTAL PRICE:", @NEWLINE( 1));
@WRITEFOR( FG(I): '  ',FG(I),'  Produce: ',
        @FORMAT(B(I),'%6.1f'), ' x Price:$',
        @FORMAT(C(I),'%6.2f'), ' = Total:$',
        @FORMAT(B(I) * P(I), '%10.2f'),
@NEWLINE( 1));
@WRITE('  TOTAL (P):',35*' ','$ ',@FORMAT(@SUM(FG(I): B(I) * P(I)),'%8.2f'),@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE('  PROFIT (P - C):',30*' ','$  ',
        @FORMAT(@SUM(FG(I): B(I) * P(I)) -
        @SUM(RF(I,J): X(I,J) * C(I)) ,'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RAW MATERIAL:
 BUT  1000.00
 CAT  4000.00
 NAP  5000.00

COST:
 BUT  7.30000
 CAT  18.2000
 NAP  12.5000

QUALITY PARAMETERS :
          OCT      VAP      VOL
 BUT  120.000  60.0000  105.000
 CAT  100.000  2.60000  3.00000
 NAP  74.0000  4.10000  12.0000

MINIMUM NEED OF EACH F.G. :
 REG  4000.00
 PRM  2000.00

MAXIMUM SELLABLE OF EACH F.G. :
 REG  8000.00
 PRM  6000.00

SELLING PRICE OF EACH F.G. :
 REG  18.4000
 PRM  22.0000

UPPER LIMITS ON QUALITY BY F.G. :
          REG      PRM
 OCT  110.000  110.000
 VAP  11.0000  11.0000
 VOL  25.0000  25.0000

LOWER LIMITS ON QUALITY BY F.G. :
          REG      PRM
 OCT  89.0000  94.0000
 VAP  8.00000  8.00000
 VOL  17.0000  17.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              48750.0
Infeasibilities:                              0.00000
Total solver iterations:                           17

 IDEAL MIXING PROGRAM:
 TOTAL COST:
 BUT     Used:  507.4 x Cost: $  7.30 = Total:$   3704.13
 CAT     Used:  492.6 x Cost: $  7.30 = Total:$   3595.87
 BUT     Used: 1410.0 x Cost: $ 18.20 = Total:$  25661.23
 CAT     Used: 2590.0 x Cost: $ 18.20 = Total:$  47138.77
 BUT     Used: 2082.6 x Cost: $ 12.50 = Total:$  26032.84
 CAT     Used: 1417.4 x Cost: $ 12.50 = Total:$  17717.16
 TOTAL (C):                                  $ 123850.00

 TOTAL PRICE:
 REG  Produce: 4000.0 x Price:$  7.30 = Total:$  73600.00
 PRM  Produce: 4500.0 x Price:$ 18.20 = Total:$  99000.00
 TOTAL (P):                                  $ 172600.00

 PROFIT (P - C):                             $  48750.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Commodity
- Gasoline
- Blend
- Refine

Source:

- Book 3
- Page 15

GOAL

A refinery processes various types of oil. Each type of oil has a different cost sheet, expressing transport conditions and cost at the source.

Resources / Products			P1	P2	P3	P4	Price
Composition	Blue Super	%	30	40	50		35.00
	Blue	%	30	10			28.00
	Yellow	%	70				22.00
Available		Barrel	3500	2200	4200	1800	-
Cost p/barrel		$	19.00	24.00	20.00	27.00	-

On the other hand, each type of oil represents a different configuration of by-products for gasoline.

To the extent that a different type of oil is used in the production of gasoline, it is possible to schedule specific octane conditions and other requirements.

These requirements imply classification of the type of gasoline obtained. Assuming the refinery works with a line of four different types of oil and wants to produce three different types of gasoline called Yellow, Blue and Super blue.

It is requested to program the mixtures of types of petroleum, according to percentages for quality limits of the gasoline types and availabilities shown below:

Blue Super:

- Not more than 30% of oil 1;
- Not less than 40% of the oil 2;
- Not more than 50% of the oil 3;

Blue:

- Not more than 30% of oil 1;
- Not less than 10% of oil 2

Yellow:

- Not more than 70% of the oil 1

```
MODEL:
SETS:
PRODUCT: COST, DEMAND;
RESOURCE:PRICE;
ROUTES( RESOURCE, PRODUCT): USAGE, PRODUCE;
ENDSETS
DATA:
!Products attributes;
PRODUCT,        COST,       DEMAND     =
PET1            19          3500
PET2            24          2200
PET3            20          4200
PET4            27          1800;
! Resources attributes;
RESOURCE,       PRICE =
GAS_SBLUE       35
GAS_BLUE        28
GAS_YELLOW      22;
!Required       PET1  PET2  PET3  PET4;
USAGE     =     30    40    50    0              ! % Composition: Gas Super Blue;
                30    10    0     0              ! % Composition: Gas Blue;
                70     0    0     0;             ! % Composition: Gas Yellow;
ENDDATA
SUBMODEL MAX3:
[OBJ]    MAX = REV1 + REV2 + REV3  - COST1 - COST2 - COST3 - COST4;
! Revenue;
REV1   + @SUM( PRODUCT(J):   -1*PRICE(1) * PRODUCE(1,J)) = 0;
REV2   + @SUM( PRODUCT(J):   -1*PRICE(2) * PRODUCE(2,J)) = 0;
REV3   + @SUM( PRODUCT(J):   -1*PRICE(3) * PRODUCE(3,J)) = 0;
! Cost;
COST1 + @SUM( RESOURCE(J): -1*COST(1)  * PRODUCE(J,1)) = 0;
COST2 + @SUM( RESOURCE(J): -1*COST(2)  * PRODUCE(J,2)) = 0;
COST3 + @SUM( RESOURCE(J): -1*COST(3)  * PRODUCE(J,3)) = 0;
COST4 + @SUM( RESOURCE(J): -1*COST(4)  * PRODUCE(J,4)) = 0;
! Restrictions associated with the amount of oil demand;
@FOR(PRODUCT(J):
        [DEM] @SUM(RESOURCE(I) : PRODUCE(I,J)) <= DEMAND(J););
! Restrictions associated with blend specifications  - Super blue Gas;
USAGE(1,1)/100 *PRODUCE(1,1) + USAGE(1,1)/100*PRODUCE(1,2) - USAGE(1,1)/100*PRODUCE(1,3) - USAGE(1,1)/
100*PRODUCE(1,4) <= 0;
USAGE(1,2)/100 *PRODUCE(1,1) + USAGE(1,2)/100*PRODUCE(1,2) - USAGE(1,2)/100*PRODUCE(1,3) - USAGE(1,2)/
100*PRODUCE(1,4) >= 0;
USAGE(1,3)/100 *PRODUCE(1,1) + USAGE(1,3)/100*PRODUCE(1,2) - USAGE(1,3)/100*PRODUCE(1,3) - USAGE(1,3)/
100*PRODUCE(1,4) <= 0;
! Restrictions associated with blend specifications - Blue Gas;
USAGE(2,1)/100 *PRODUCE(2,1) + USAGE(2,1)/100*PRODUCE(2,2) - USAGE(2,1)/100*PRODUCE(2,3) - USAGE(2,1)/
100*PRODUCE(2,4) <= 0;
USAGE(2,2)/100 *PRODUCE(2,1) + USAGE(2,2)/100*PRODUCE(2,2) - USAGE(2,2)/100*PRODUCE(2,3) - USAGE(2,2)/
100*PRODUCE(2,4) >= 0;
! Restrictions associated with blend specifications - Yellow Gas;
USAGE(3,1)/100 *PRODUCE(3,1) + USAGE(3,1)/100*PRODUCE(3,2) - USAGE(3,1)/100*PRODUCE(3,3) - USAGE(3,1)/
100*PRODUCE(3,4) <= 0;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  COMPOSITION (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (barrel):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  COST OF OIL (per barrel):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SALE PRICE (per barrel):", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ROUTES(I,J) | PRODUCE(I,J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J), '-4s'), ' , ',
        @FORMAT(RESOURCE( I),'-8s'),': ',
        @FORMAT(PRODUCE( I,J), '%4.0f'),'Barrel x Price:$',
        @FORMAT(PRICE( I),'%4.2f'), ' = Revenue:$',
        @FORMAT(PRICE( I) * PRODUCE(I, J),'%9.2f'), ' - Cost:$',
        @FORMAT(COST( J) * PRODUCE(I, J),'%7.2f'), ' = Profit: $',
        @FORMAT(PRICE( I) * PRODUCE(I, J) - COST( j) * PRODUCE(I, J),'%8.2f'),
@NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COMPOSITION (%):
                 PET1      PET2      PET3      PET4
 GAS_SBLUE   30.00000  40.00000  50.00000  0.000000
 GAS_BLUE    30.00000  10.00000  0.000000  0.000000
 GAS_YELLOW  70.00000  0.000000  0.000000  0.000000

DEMAND (barrel):
 PET1        3500.000
 PET2        2200.000
 PET3        4200.000
 PET4        1800.000

COST OF OIL (per barrel):
 PET1        19.00000
 PET2        24.00000
 PET3        20.00000
 PET4        27.00000

SALE PRICE (per barrel):
 GAS_SBLUE   35.00000
 GAS_BLUE    28.00000
 GAS_YELLOW  22.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                                    155200.0
Infeasibilities:                                                                    0.000000

IDEAL PLANNING PROGRAM:
PET1, GAS_SBLUE: 3500 barrel x Price: $35.00 = Revenue: $122500.00 - Cost: $66500.00 = Profit: $56000.00
PET2, GAS_SBLUE: 2200 barrel x Price: $35.00 = Revenue: $ 77000.00 - Cost: $52800.00 = Profit: $24200.00
PET3, GAS_SBLUE: 4200 barrel x Price: $35.00 = Revenue: $147000.00 - Cost: $84000.00 = Profit: $63000.00
PET4, GAS_SBLUE: 1500 barrel x Price: $35.00 = Revenue: $ 52500.00 - Cost: $40500.00 = Profit: $12000.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Commodity
- Gasoline
- Blend
- Refine

Source:

- Book 2
- Chapter 14.35

GOAL

A refinery produces three types of gasoline: Green, Blue and Yellow. Each type requires Pure Petrol, Octane and Additives in the available quantities of 9600000, 4800000 and 2200000 liters per week respectively.

- One liter of green gasoline requires 0.22 liters of pure gasoline, 0.50 liters of octane and 0.28 liters of additives.
- One liter of blue gasoline requires 0.52 liters of pure gasoline, 0.34 liters of octane and 0.14 liters of additives.
- One liter of yellow petrol requires 0.74 liters of pure gasoline, 0.20 liters of octane and 0.06 liters of additives.

Refinery planning stipulated that the amount of yellow gasoline should be at least 16 times the amount of green gasoline and the amount of blue gasoline should be equal to 600,000 liters per week.

The company knows that every liter of green, blue and yellow gasoline has a profit margin of $ 0.30, $ 0.25 and $ 0.20 respectively.

The goal is to determine the production schedule that maximizes the total contribution margin for profit. The data necessary to elaborate the model are shown below:

Resources / Products			Green	Blue	Yellow	Available
Composition	Pure Gasoline	L	0.22	0.52	0.74	9,600,000
	Octane	L	0.50	0.34	0.20	4,800,000
	Additive	L	0.28	0.14	0.06	2,200,000
Planned Production		L	G	≤ 600,000	16 x G	16,600,000
Profit		$	0.30	0.25	0.20	

```
MODEL:
SETS:
PRODUCT: PROFIT, PRODUCE;
RESOURCE:AVAILABLE;
ROUTES( RESOURCE, PRODUCT): FORMULA;
ENDSETS
DATA:
! Resource attributes;
RESOURCE,        AVAILABLE    =
PURE_GAS         9600000
OCTANE           4800000
ADDITIVE         2200000;
! Products attributes;
PRODUCT,         PROFIT       =
GAS_GREEN        0.30
GAS_BLUE         0.25
GAS_YELLOW       0.20;
! Required (L)    GAS_GREEN    GAS_BLUE    GAS_YELLOW;
FORMULA  =        0.22         0.52        0.74        ! Pure gasoline;
                  0.50         0.34        0.20        ! Octane;
                  0.28         0.14        0.06;       ! Additive;
ENDDATA
SUBMODEL MAX4:
[OBJ] MAX = @SUM(PRODUCT(J): PROFIT(J) * PRODUCE(J));
! The available constraints;
@FOR(RESOURCE(I):
        @SUM(PRODUCT(J): FORMULA(I,J) * PRODUCE(J)) <= AVAILABLE(I););
! Green gasoline should be 1/16 of the yellow gasoline;
PRODUCE(1) >= PRODUCE(3)/16;
! Blue gasoline must be ≤ 600000 liters ;
PRODUCE(2)  = 600000;
! Yellow gasoline should be 16 times the amount of green gasoline;
PRODUCE(3)  = PRODUCE(1)*16;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (L):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (L):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute Sub-model;
@SOLVE(MAX4);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITE("  PRODUCED: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J): '  . ',
        @FORMAT(PRODUCT( J),'-10s'),' ',
        @FORMAT(PRODUCE( J), '%8.0f'),' L x Profit: $',
        @FORMAT(PROFIT( J),'%4.2f'), ' = Total: $',
        @FORMAT(PROFIT( J)  * PRODUCE( J),'%10.2f'),
@NEWLINE(1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (L):
                  GAS_GREEN          GAS_BLUE        GAS_YELLOW
 PURE_GAS      0.2200000000      0.5200000000      0.7400000000
 OCTANE        0.5000000000      0.3400000000      0.2000000000
 ADDITIVE      0.2800000000      0.1400000000      0.0600000000

AVAILABLE (L):
 PURE_GAS      9600000.000
 OCTANE        4800000.000
 ADDITIVE      2200000.000

PROFIT:
 GAS_GREEN     0.300000000
 GAS_BLUE      0.250000000
 GAS_YELLOW    0.200000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                   2845522.388
Infeasibilities:                                   0.000000000

IDEAL PLANNING PROGRAM:
PRODUCED:
. GAS_GREEN    770149 L  x  Profit: $0.30 = Total: $ 231044.78
. GAS_BLUE     600000 L  x  Profit: $0.25 = Total: $ 150000.00
. GAS_YELLOW 12322388 L  x  Profit: $0.20 = Total: $2464477.61
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Gasoline
- Blend

Source:

- Book 2
- Chapter 14.22

GOAL

A distribution company sells ordinary and special gasoline. Each liter of ordinary gasoline is sold at $2.10 and must have at least 90 octane. Each liter of special gasoline is sold for $2.50 and needs to have at least 97 octane.

These types of gasoline are obtained by mixing three different types of gasolines, as shown in the following table:

Resources / Products	Cost p/L	Octane	Special	Common	Available (L)
Gas 1	$1.725	100	0	1.0	150000
Gas 2	$1.575	87	0.67	0.33	350000
Gas 3	$1.775	110	0.83	0	300000
Minimum Required Octane			97	90	-
Demand p/L			450,000	300,000	-
Price p/L			$2.50	$2.10	-

The company received an order for 300,000 liters of ordinary gasoline and 450,000 liters of special gasoline. What should be the ideal mix to fulfill the request maximizing profit.

```
MODEL:
SETS:
PRODUCT: PRICE, DEMAND, MINREQ, PRODUCE;
RESOURCE:AVAILABLE, OCTANE, COST;
ROUTES( RESOURCE, PRODUCT): FORMULA;
ENDSETS
DATA:
! Resource attributes;
RESOURCE,         AVAILABLE,        OCTANE,     COST  =
GAS1              150000            100         1.725
GAS2              350000            87          1.575
GAS3              300000            110         1.775;
! Products attributes;
PRODUCT,          PRICE,            DEMAND      MINREQ      =
GAS_SPECIAL       2.50              450000      97
GAS_COMMON        2.10              300000      90;
! Required        GAS_SPECIAL       GAS_COMMON;
FORMULA  =        0.00              1.00        ! GAS1;
                  0.67              0.33        ! GAS2;
                  0.833333          0.00 ;      ! GAS3;
ENDDATA
SUBMODEL MAX5:
[OBJ] MAX = PRICE_T - COST_T;
! Calculation of total price;
PRICE_T =     @SUM(PRODUCT(J): PRICE(J) * PRODUCE(J));
COST_T  =     COST(1) * (AVAILABLE(1) * FORMULA(1,1) + AVAILABLE(1) * FORMULA(1,2)) +
              COST(2) * (AVAILABLE(2) * FORMULA(2,1) + AVAILABLE(2) * FORMULA(2,2)) +
              COST(3) * (AVAILABLE(3) * FORMULA(3,1) + AVAILABLE(3) * FORMULA(3,2)) ;
! The Demand constraints;
PRODUCE(1) = DEMAND(1);  PRODUCE(2) = DEMAND(2);
ENDSUBMODEL;
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (L):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (L):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND: (L)", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  COST P/LITER:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  PRICE P/LITER:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  OCTANE:", @NEWLINE( 1));
@TABLE(OCTANE);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM REQUIRED OCTANE:", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX5);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1), "  MIXED: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I): '  + ',
        @FORMAT(PRODUCT( I),'-11s'),':',
        @FORMAT(PRODUCE( I), '%8.0f'),' L x Price: $',
        @FORMAT(PRICE( I),'%4.2f'), ' = Total: $',
        @FORMAT(PRICE( I)  * PRODUCE( I),'%10.2f'),
@NEWLINE(1));
@WRITE("  - TOTAL COST: ", 35*' ','$',@FORMAT(COST_T,'%10.2f'),@NEWLINE( 1));
@WRITE("  = PROFIT: ", 39*' ', '$',@FORMAT(PRICE_T - COST_T,'%10.2f'),@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (L):
            GAS_SPECIAL   GAS_COMMON
 GAS1        0.0000000     1.000000
 GAS2        0.6700000     0.330000
 GAS3        0.8333330     0.000000

AVAILABLE (L):
 GAS1        150000.0
 GAS2        350000.0
 GAS3        300000.0

DEMAND: (L)
 GAS_SPECIAL  450000.0
 GAS_COMMON   300000.0

COST P/LITER:
 GAS1        1.725000
 GAS2        1.575000
 GAS3        1.775000

PRICE P/LITER:
 GAS_SPECIAL  2.500000
 GAS_COMMON   2.100000

OCTANE:
 GAS1        100.0000
 GAS2        87.00000
 GAS3        110.0000

MINIMUM REQUIRED OCTANE:
 GAS_SPECIAL  97.00000
 GAS_COMMON   90.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                   501250.2
Infeasibilities:                                   0.000000

IDEAL PLANNING PROGRAM:
MIXED:
+ GAS_SPECIAL:  450000 L x Price: $2.50 = Total: $1125000.00
+ GAS_COMMON :  300000 L x Price: $2.10 = Total: $ 630000.00
- TOTAL COST:                                    $1253749.82
= PROFIT:                                        $ 501250.18
```

BLOCK 10

Block: SCHEDULE

How to designate sections of a highway to be built or designate work groups so as to obtain the lowest possible cost?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- **Schedule**
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

SCHEDULE STAFF

One part of the management of most service facilities is the scheduling or staffing of personnel. That is, deciding how many people to use on what shifts.

This problem exists in staffing the information operators department of a telephone company, a toll plaza, a large hospital, and, in general, any facility that must provide service to the public.

The solution process consists of at least three parts:

1. Develop good forecasts of the number of personnel required during each hour of the day or each day of the week during the scheduling period.

2. Identify the possible shift patterns, which can be worked based on the personnel available and work agreements and regulations. A particular shift pattern might be to work Tuesday through Saturday and then be off two days.

3. Determine how many people should work each shift pattern, so cost are minimized and the total number of people on duty during each time period satisfies the requirements determined in (1).

All three of these steps are difficult. LP can help in solving step 3.

One of the first published accounts of using optimization for staff scheduling was by Edie (1954). He developed a method for staffing tollbooths for the New York Port Authority.

Though old, Edie's discussion is still very pertinent and thorough. His thoroughness is illustrated by his summary (p. 138):

- A trial was conducted at the Lincoln Tunnel…
- Each toll collector was given a slip showing his booth assignments and relief periods and instructed to follow the schedule strictly…
- At no times did excessive backups occur…
- The movement of collectors and the opening and closing of booths took place without the attention of the toll sergeant.
- At times, the number of booths were slightly excessive, but not to the extent previously...
- Needless to say, there is a good deal of satisfaction...

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Straight

Source:

- Book 1
- Page 106

GOAL

Several companies have submitted a proposal to build four stretches of a road. Assemble the model of highway designation at the lowest cost, considering that each company can only do one stretch. The costs offered by the companies are in the table below.

Resources / Products			Straight 1	Straight 2	Straight 3	Straight 4
Companies	A	$	500.00	700.00	300.00	200.00
	B	$	450.00	1000.00	450.00	250.00
	C	$	650.00	800.00	500.00	320.00
	D	$	550.00	950.00	480.00	280.00

```
MODEL:
SETS:
 PRODUCT :;
 RESOURCE: ;
 RXP( RESOURCE, PRODUCT) : USAGE, PRODUCE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE  =
 CIA_1
 CIA_2
 CIA_3
 CIA_4;
! Products attributes;
 PRODUCT    =
 STRAIGHT1
 STRAIGHT2
 STRAIGHT3
 STRAIGHT4 ;

! Required          CIA_1       CIA_2       CIA_3       CIA_4;
 USAGE       =      500         700         300         200         ! STRAIGHT1;
                    450         1000        450         250         ! STRAIGHT2;
                    650         800         500         320         ! STRAIGHT3;
                    550         950         480         280;        ! STRAIGHT4;
ENDDATA
SUBMODEL  MIN1:
[OBJ] MIN = @SUM( RXP(I,C): USAGE( I, C) * PRODUCE( I, C));
! Selects the company;
@FOR( PRODUCT(I):
        @SUM(RESOURCE(C): PRODUCE(I,C)) = 1);
! Select the stretch of highway;
@FOR( PRODUCT(I):
        @SUM(RESOURCE(C): PRODUCE(C, I)) = 1);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "   COST:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J)| PRODUCE(I,J) #GT# 0: '  ',
        @FORMAT(RESOURCE( I), '-5s'),'   Selected to do  ',
        @FORMAT(PRODUCT( J), '-10S'),'  by  Cost: $',
        @FORMAT(USAGE(I,J), '%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
 COST:
           STRAIGHT1  STRAIGHT2  STRAIGHT3  STRAIGHT4
 CIA_1      500.0000   700.0000   300.0000   200.0000
 CIA_2      450.0000   1000.000   450.0000   250.0000
 CIA_3      650.0000   800.0000   500.0000   320.0000
 CIA_4      550.0000   950.0000   480.0000   280.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              1830.000
Infeasibilities:                              0.000000

IDEAL PLANNING PROGRAM:
CIA_1   Selected to do  STRAIGHT3   by  Cost: $ 300.00
CIA_2   Selected to do  STRAIGHT1   by  Cost: $ 450.00
CIA_3   Selected to do  STRAIGHT2   by  Cost: $ 800.00
CIA_4   Selected to do  STRAIGHT4   by  Cost: $ 280.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Staff
- Shift
- Allocation

GOAL

A service delivery company needs to allocate watchmen on several clients, but it has been found that there is a variation between the needs of each day of the week as can be observed below:

Required / Day	MON	TUE	WED	THU	FRI	SAT	SUN
Worker	20	30	24	44	12	22	25

It is also known that the work week is five days in a row and two days off. We ask for the best and least personal allocation scheme

```
MODEL:
SETS:
DAYS: REQUIRED, START, ONDUTY;
ENDSETS
DATA:
!  Days attributes;
DAYS,  REQUIRED    =
MON           20
TUE           30
WED           24
THU           44
FRI           12
SAT           22
SUN           25;
ENDDATA
SUBMODEL MIN2:
[OBJ]  MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
        ONDUTY(J) = @SUM( DAYS( I) | I #LE# 5: START( @WRAP( J - I + 1, 7)));
        ONDUTY(J) >= REQUIRED( J) );
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  REQUIRED:", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( DAYS(D): '  ',
        @FORMAT(DAYS( D),'-4S'),' ', "Required: ",
        @FORMAT(REQUIRED(D),'%2.0f'), ' Worker  ', 'Start:',
        @FORMAT(START(D),'%3.0f'), '  On duty:', ONDUTY(D), '  Surplus: ',
        @FORMAT(ONDUTY(D) - REQUIRED(D),'%2.0f'), @NEWLINE( 1));
@WRITE('  Worker total:',19*' ',@FORMAT(OBJ,'%3.0f'),@NEWLINE(1));
! Bar chart of required vs. actual staffing;
 @CHARTBAR(
  'Staffing Schedule',                   !Chart title;
  'Day',                          !X-Axis label;
  'Employees',                    !Y-Axis label;
  'Employees Required',           !Legend 1;
  REQUIRED,             !Attribute 1;
  'Employees On Duty',            !Legend 2;
  ONDUTY                          !Attribute 2;);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

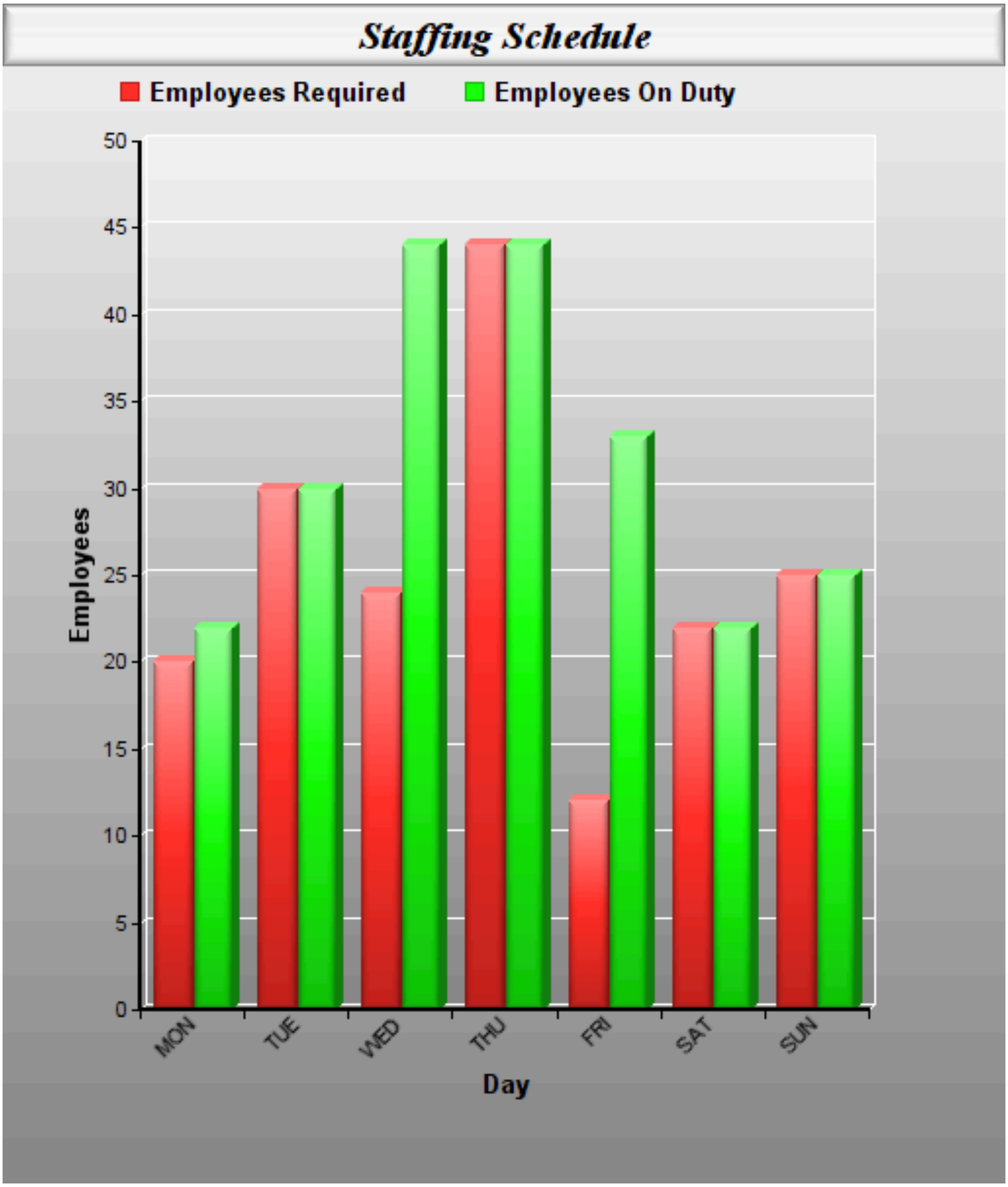
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED (worker):
 MON  20.00000
 TUE  30.00000
 WED  24.00000
 THU  44.00000
 FRI  12.00000
 SAT  22.00000
 SUN  25.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                          44.00000
Infeasibilities:                                          0.000000

IDEAL PLANNING PROGRAM:
MON  Required: 20 Worker  Start: 11  On duty:22.00000  Surplus:  2
TUE  Required: 30 Worker  Start:  8  On duty:30.00000  Surplus:  0
WED  Required: 24 Worker  Start: 14  On duty:44.00000  Surplus: 20
THU  Required: 44 Worker  Start:  0  On duty:44.00000  Surplus:  0
FRI  Required: 12 Worker  Start:  0  On duty:33.00000  Surplus: 21
SAT  Required: 22 Worker  Start:  0  On duty:22.00000  Surplus:  0
SUN  Required: 25 Worker  Start: 11  On duty:25.00000  Surplus:  0
Worker total:                    44
```


Staffing Schedule
Employees Required
Employees On Duty
Employees
Day
MON
TUE
WED
THU
FRI
SAT
SUN
0
5
10
15
20
25
30
35
40
45
50

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Staff
- Allocation

GOAL

A post office needs a different number of employees, depending on the day of the week. For union demands, each worker works five consecutive days and rests two.

Required / Day	MON	TUE	WED	THU	FRI	SAT	SUN
Worker	20	12	18	16	14	14	12

Formulate the problem so that the number of employees hired is the minimum necessary to meet the needs of the agency.

```
MODEL:
SETS:
DAYS: REQUIRED, START, ONDUTY;
ENDSETS
DATA:
DAYS,        REQUIRED  =
MON          20
TUE          12
WED          18
THU          16
FRI          14
SAT          14
SUN          12;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
      ONDUTY(J) = @SUM( DAYS( I) | I #LE# 5: START( @WRAP( J - I + 1, 7)));
      ONDUTY(J) >= REQUIRED( J));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  REQUIRED:", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 2));
! Execute Sub-model;
@SOLVE(MIN3);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( DAYS(D): '  ',
      @FORMAT(DAYS( D),'-3s'),' ', "Required: ",
      @FORMAT(REQUIRED(D),'%2.0f'), ' Worker, ', 'Start:',
      @FORMAT(START(D),'%2.0f'), ' On duty:',
      @FORMAT(ONDUTY(D),'%3.0f'), ' Surplus:',
      @FORMAT(ONDUTY(D) - REQUIRED(D),'%2.0f'),
@NEWLINE( 1));
@WRITE('  Worker total:',17*' ',@FORMAT(OBJ,'%3.0f'),@NEWLINE(1));
! Bar chart of required vs. actual staffing;
 @CHARTBAR(
  'Staffing Schedule',              !Chart title;
  'Day',                     !X-Axis label;
  'Employees',               !Y-Axis label;
  'Employees Required',      !Legend 1;
  REQUIRED,          !Attribute 1;
  'Employees On Duty',       !Legend 2;
  ONDUTY                     !Attribute 2;);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED (worker):
 MON  20.00000
 TUE  12.00000
 WED  18.00000
 THU  16.00000
 FRI  14.00000
 SAT  14.00000
 SUN  12.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:

Global optimal solution found.
Objective value:                                 22.66667
Infeasibilities:                                 0.000000

IDEAL PLANNING PROGRAM:
MON Required: 20 Worker, Start: 9 On duty: 20 Surplus: 0
TUE Required: 12 Worker, Start: 2 On duty: 17 Surplus: 5
WED Required: 18 Worker, Start: 1 On duty: 18 Surplus: 0
THU Required: 16 Worker, Start: 5 On duty: 16 Surplus: 0
FRI Required: 14 Worker, Start: 0 On duty: 16 Surplus: 2
SAT Required: 14 Worker, Start: 7 On duty: 14 Surplus: 0
SUN Required: 12 Worker, Start: 0 On duty: 12 Surplus: 0
Worker total:                  23
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Staff
- Allocation

Source:

- Book 4
- Page 82

GOAL

Suppose you run the popular Pluto Dogs dog cart which is open seven days a week.

You hire employees for a five-day work week with two consecutive days. Each employee receives the same weekly salary.

Some days of the week are busier than others and, based on past experience, you know how many workers are needed on a particular day of the week. In particular, your forecast calls these personnel requirements:

Required / Day	MON	TUE	WED	THU	FRI	SAT	SUN
Worker	20	12	18	16	14	14	12

You need to determine how many employees start each day of the week in order to minimize the total number of employees while still meeting or exceeding the staffing requirements every day of the week.

```
MODEL:
SETS:
DAYS: REQUIRED, START, ONDUTY;
ENDSETS
DATA:
DAYS,         REQUIRED   =
MON           20
TUE           16
WED           13
THU           16
FRI           19
SAT           14
SUN           12;
ENDDATA
SUBMODEL MIN4:
[OBJ] MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
        ONDUTY(J) = @SUM( DAYS( I) | I #LE# 5: START( @WRAP( J - I + 1, 7)));
        ONDUTY(J) >= REQUIRED( J));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! data block;
@WRITE("  REQUIRED (worker):", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN4);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( DAYS(D): '  ',
        @FORMAT(DAYS( D),'-3s'), " Required: ",
        @FORMAT(REQUIRED(D), '%3.0f'),' Worker  ', 'Start:',
        @FORMAT(START(D),'%3.0f'), '  On duty:',
        @FORMAT(ONDUTY(D),'%3.0f'), '  Surplus: ',
        @FORMAT(ONDUTY(D) - REQUIRED(D),'%2.0f'),
@NEWLINE( 1));
@WRITE('  Worker total:',19*' ',@FORMAT(OBJ,'%3.0f'),@NEWLINE(1));
! Bar chart of required vs. actual staffing;
 @CHARTBAR(
  'Staffing Schedule',                 !Chart title;
  'Day',                         !X-Axis label;
  'Employees',                   !Y-Axis label;
  'Employees Required',          !Legend 1;
  REQUIRED,            !Attribute 1;
  'Employees On Duty',           !Legend 2;
  ONDUTY                         !Attribute 2;);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED (worker):
 MON  20.00000
 TUE  16.00000
 WED  13.00000
 THU  16.00000
 FRI  19.00000
 SAT  14.00000
 SUN  12.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    22.00000
Infeasibilities:                                    0.000000

IDEAL PLANNING PROGRAM:
MON Required:  20 Worker  Start:  8  On duty: 20  Surplus:  0
TUE Required:  16 Worker  Start:  2  On duty: 16  Surplus:  0
WED Required:  13 Worker  Start:  0  On duty: 13  Surplus:  0
THU Required:  16 Worker  Start:  6  On duty: 16  Surplus:  0
FRI Required:  19 Worker  Start:  3  On duty: 19  Surplus:  0
SAT Required:  14 Worker  Start:  3  On duty: 14  Surplus:  0
SUN Required:  12 Worker  Start:  0  On duty: 12  Surplus:  0
Worker total:                    22
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Staff
- Job
- Allocation

GOAL

In this model, there are six jobs that can be done on one machine.The machine can only run one job at a time. Each of the papers has an expiration date.

Resources / Job		J1	J2	J3	J4	J5	J6
Due Date	Day	9	3	6	5	7	2
Machine Time	hr	5	2	4	3	1	2
Price	$	9.00	2.00	4.00	2.00	4.00	6.00

If we can not complete a job by the due date, we will not. Our goal is to maximize the total value of the selected papers.

```
MODEL:
SETS:
! There are six jobs, each of which has a due date,
  Processing time, value, and a flag variable Y indicating whether the job was selected;
 JOB/1..6/:      ! Each job has a...;
  DD,            ! Due date;
  PT,            ! Machine time;
  VAL,           ! Job price;
  Y;             ! 1 if job is selected;
ENDSETS
DATA:
 VAL             = 9 2 4 2 4 6;
 DD              = 9 3 6 5 7 2;
 PT              = 5 2 4 3 1 2;
ENDDATA
SUBMODEL MAX5:
! Maximize the total value of the works to be performed;
 [OBJ] MAX = TVAL;
          TVAL = @SUM( JOB: VAL * Y);
! For the selected works, we do on the due date of the request;
 @FOR( JOB( J):
         ! Only jobs with previous expiration dates can precede job J, and work must be completed
         according to their maturity dates;
         [CON] @SUM( JOB( I)| DD( I) #LT# DD( J) #OR#(  DD( I) #EQ# DD( J) #AND# I #LE# J):  PT( I) * Y( I)) <= DD( J);
         ! Define the Y as binary variable;
         @BIN( Y););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data Block;
@WRITE("  JOB PRICE:", @NEWLINE( 1));
@TABLE(VAL);
@WRITE(" ", @NEWLINE( 1), "  DUE DATE:", @NEWLINE( 1));
@TABLE(DD);
@WRITE(" ", @NEWLINE( 1), "  MACHINE TIME:", @NEWLINE( 1));
@TABLE(PT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX5);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( JOB(D)|Y(D) #GT# 0: '  Due date:',
         @FORMAT(DD( D),'%1.0f'),' ', " Machine Time:",
         @FORMAT(PT( D),'%1.0f'), '  Price: $',
         @FORMAT(VAL(D) * Y(D), '%4.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
JOB PRICE:
 1  9.000000
 2  2.000000
 3  4.000000
 4  2.000000
 5  4.000000
 6  6.000000

DUE DATE:
 1  9.000000
 2  3.000000
 3  6.000000
 4  5.000000
 5  7.000000
 6  2.000000

MACHINE TIME:
 1  5.000000
 2  2.000000
 3  4.000000
 4  3.000000
 5  1.000000
 6  2.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                19.00000
Objective bound:                19.00000
Infeasibilities:                0.000000


IDEAL PLANNING PROGRAM (JOB SELECTED):
Due date:9  Machine Time:5  Price: $9.00
Due date:7  Machine Time:1  Price: $4.00
Due date:2  Machine Time:2  Price: $6.00
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Shift
- Job
- Allocation

Source:

- Book 5
- Page 114

GOAL

The Northeast Tollway out of Chicago has a toll plaza with the following staffing demands during each 24-hour period:

SEQ	Hours			Collectors Need
1	0 A.M.	To	6 A.M.	2
2	6 A.M.	To	10 A.M.	8
3	10 A.M.	To	Noon	4
4	Noon	To	4 P.M.	3
5	4 P.M.	To	6 P.M	6
6	6 P.M.	To	10 P.M	5
7	10 PM	To	12 Midnight	3

Each collector works four hours, is off one hour, and then works another four hours. A collector can be started at any hour. Assuming the objective is to minimize the number of collectors hired, how many collectors should start work each hour?

```
MODEL:
! 24 hour shift scheduling;
! Each shift is4 hours on, 1 hour off, 4 hours on;
SETS:
   HOUR/1..24/: START, REQUIRED, ONDUTY;
ENDSETS
DATA:
REQUIRED  =  2 2 2 2 2 2 8 8 8 8 4 4 3 3 3 3 6 6 5 5 5 5 3 3;
ENDDATA
SUBMODEL MIN6:
[OBJ] MIN = @SUM( HOUR(I): START(I));
! People on duty in hour I are those who  started 9 or less hours earlier, but not 5;
@FOR( HOUR( I):
        ONDUTY = @SUM(HOUR(J)|(J#LE#9)#AND#(J#NE#5): START(@WRAP((I-J+1),24)));
        ONDUTY >= REQUIRED(I);
        @GIN(START(I)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  REQUIRED (worker):", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN6);
! Solution report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( HOUR(D) | START(D) #GT# 0: '  Hour:',
        @FORMAT( HOUR( D),'3s'),', ', "Required:",
        @FORMAT(REQUIRED(D),'%2.0f'), ' Collectors, ', 'Start work at:',
        @FORMAT(START(D),'%2.0f'), '  On duty:',
        @FORMAT(ONDUTY(D),'%2.0f'), '  Surplus:',
        @FORMAT(ONDUTY(D) - REQUIRED(D),'%2.0f'),
@NEWLINE( 1));
@TEXT() = @WRITE('  Worker total:',35*' ',@FORMAT(OBJ,'%2.0f'),@NEWLINE(1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED (hr,worker):
 1   2.000000
 2   2.000000
 3   2.000000
 4   2.000000
 5   2.000000
 6   2.000000
 7   8.000000
 8   8.000000
 9   8.000000
 10  8.000000
 11  4.000000
 12  4.000000
 13  3.000000
 14  3.000000
 15  3.000000
 16  3.000000
 17  6.000000
 18  6.000000
 19  5.000000
 20  5.000000
 21  5.000000
 22  5.000000
 23  3.000000
 24  3.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                16.00000
Objective bound:                                                16.00000
Infeasibilities:                                                0.000000

IDEAL PLANNING PROGRAM:
Hour:  2, Required: 2 Collectors, Start work at: 5  On duty: 6  Surplus: 4
Hour:  3, Required: 2 Collectors, Start work at: 1  On duty: 6  Surplus: 4
Hour:  4, Required: 2 Collectors, Start work at: 1  On duty: 7  Surplus: 5
Hour:  5, Required: 2 Collectors, Start work at: 1  On duty: 8  Surplus: 6
Hour:  6, Required: 2 Collectors, Start work at: 1  On duty: 4  Surplus: 2
Hour: 12, Required: 4 Collectors, Start work at: 1  On duty: 4  Surplus: 0
Hour: 14, Required: 3 Collectors, Start work at: 1  On duty: 3  Surplus: 0
Hour: 15, Required: 3 Collectors, Start work at: 1  On duty: 3  Surplus: 0
Hour: 16, Required: 3 Collectors, Start work at: 2  On duty: 4  Surplus: 1
Hour: 17, Required: 6 Collectors, Start work at: 1  On duty: 6  Surplus: 0
Hour: 18, Required: 6 Collectors, Start work at: 1  On duty: 6  Surplus: 0
Worker total:                                   16
```

7

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Designation
- Shift
- Job
- Allocation

Source:
- Book 3
- Chapter 2.3.9

GOAL

A hospital works with a 24/7 customer service. The requirements are distributed according which shift.

TIMETABLE
- SHIFT1 08-12 hr
- SHIFT2 12-16 hr
- SHIFT3 16-20 hr
- SHIFT4 20-00 hr
- SHIFT5 00-04 hr
- SHIFT6 04-08 hr

Minimize the number of nurses at work for 24 Hours

```
MODEL:
SETS:
DAYS: REQUIRED, START, ONDUTY;
ENDSETS
DATA:
! Days required                    Timetable ;
DAYS, REQUIRED   =
SHIFT1       50                    ! 8-12;
SHIFT2       60                    ! 12-16;
SHIFT3       50                    ! 16-20;
SHIFT4       40                    ! 20-0;
SHIFT5       30                    ! 0-4 ;
SHIFT6       20;                   ! 4-8;
ENDDATA
SUBMODEL MIN7:
[OBJ] MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
        ONDUTY(J) = @SUM( DAYS( I) | I #LE# 5: START( @WRAP( J - I + 1, 6)));
        ONDUTY(J) >= REQUIRED( J));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! data block;
@WRITE("  REQUIRED:", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN7);
! Solution report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( DAYS(D): '  ',
        @FORMAT(DAYS( D),'-6s'),' ', "Required:",
        @FORMAT(REQUIRED(D),'%3.0f'), ' Nurses  ', 'Start:',
        @FORMAT(START(D),'%3.0f'), '  On duty:',
        @FORMAT(ONDUTY(D),'%3.0f'), '  Surplus: ',
        @FORMAT(ONDUTY(D) - REQUIRED(D),'%2.0f'),
@NEWLINE( 1));
@WRITE('  Nurses total:',22*' ',@FORMAT(OBJ,'%2.0f'),@NEWLINE(1));
! Bar chart of required vs. actual staffing;
 @CHARTBAR(
  'Staffing Schedule',              !Chart title;
  'Day',                       !X-Axis label;
  'Employees',                 !Y-Axis label;
  'Employees Required',        !Legend 1;
  REQUIRED,           !Attribute 1;
  'Employees On Duty',         !Legend 2;
  ONDUTY                       !Attribute 2;);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED NURSES PER SHIFT:
 SHIFT1  50.00000
 SHIFT2  60.00000
 SHIFT3  50.00000
 SHIFT4  40.00000
 SHIFT5  30.00000
 SHIFT6  20.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                         60.00000
Infeasibilities:                                         0.000000

IDEAL PLANNING PROGRAM:
SHIFT1 Required: 50 Nurses  Start: 40  On duty: 50  Surplus:  0
SHIFT2 Required: 60 Nurses  Start: 10  On duty: 60  Surplus:  0
SHIFT3 Required: 50 Nurses  Start:  0  On duty: 50  Surplus:  0
SHIFT4 Required: 40 Nurses  Start: 10  On duty: 60  Surplus: 20
SHIFT5 Required: 30 Nurses  Start:  0  On duty: 60  Surplus: 30
SHIFT6 Required: 20 Nurses  Start:  0  On duty: 20  Surplus:  0
Nurses total:                     60
```

8

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Designation
- Shift
- Job
- Allocation

Source:

- Book 2
- Chapter 14.25

GOAL

The maintenance of a theme park operates 24 Hours a day. The work day is 8 hours a day and shifts occur every 4 hours.

TIMETABLE

- SHIFT1 24-04 hr
- SHIFT2 04-08 hr
- SHIFT3 08-12 hr
- SHIFT4 12-16 hr
- SHIFT5 16-20 hr
- SHIFT6 20-24 hr

The maintenance supervisor wants to define the minimum number of employees at each shift in order to meet the minimum needs required, thus determining the minimum number of employees.

```
MODEL:
SETS:
DAYS: REQUIRED, START, ONDUTY;
ENDSETS
DATA:
! Days required                    Timetable;
DAYS ,      REQUIRED   =
SHIFT1      90                     ! 24-04;
SHIFT2      215                    ! 04-08;
SHIFT3      250                    ! 08-12;
SHIFT4      165                    ! 12-16;
SHIFT5      300                    ! 16-20;
SHIFT6      125;                   ! 20-24;
ENDDATA
SUBMODEL MIN8:
[OBJ] MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
      ONDUTY(J) = @SUM( DAYS( I) | I #LE# 5: START( @WRAP( J - I + 1, 6)));
      ONDUTY(J) >= REQUIRED( J));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  REQUIRED WORKER PER SHIFT:", @NEWLINE( 1));
@TABLE(REQUIRED);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN8);
! Solution report;
@WRITE(" ", @NEWLINE( 1));
@WRITE("  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( DAYS(D): '  ', DAYS( D),' ', "Required: ",
      @FORMAT(REQUIRED(D),'%4.0f'), ' Worker  ', 'Start:',
      @FORMAT(START(D),'%4.0f'), '  On duty:',
      @FORMAT(ONDUTY(D),'%4.0f'), '  Surplus: ',
      @FORMAT(ONDUTY(D) - REQUIRED(D),'%3.0f'),
@NEWLINE( 1));
@WRITE('  Worker total:',24*' ',@FORMAT(OBJ,'%2.0f'),@NEWLINE(1));
! Bar chart of required vs. actual staffing;
 @CHARTBAR(
  'Staffing Schedule',              !Chart title;
  'Shift',                    !X-Axis label;
  'Worker',                   !Y-Axis label;
  'Worker Required',          !Legend 1;
  REQUIRED,          !Attribute 1;
  'Worker On Duty',           !Legend 2;
  ONDUTY                      !Attribute 2;);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN7);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
REQUIRED WORKER PER SHIFT:
 SHIFT1  90.00000
 SHIFT2  215.0000
 SHIFT3  250.0000
 SHIFT4  165.0000
 SHIFT5  300.0000
 SHIFT6  125.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                            300.0000
Infeasibilities:                                            0.000000

IDEAL PLANNING PROGRAM:
SHIFT1 Required:   90 Worker  Start: 175  On duty: 175  Surplus:  85
SHIFT2 Required:  215 Worker  Start: 125  On duty: 300  Surplus:  85
SHIFT3 Required:  250 Worker  Start:   0  On duty: 300  Surplus:  50
SHIFT4 Required:  165 Worker  Start:   0  On duty: 300  Surplus: 135
SHIFT5 Required:  300 Worker  Start:   0  On duty: 300  Surplus:   0
SHIFT6 Required:  125 Worker  Start:   0  On duty: 125  Surplus:   0
Worker total:                     300
```

BLOCK 11 Block: CUTTING

How to optimize cuts of Paper, Iron or Wood Plates, Tubes, Carpe, etc., in a way to minimize losses?

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Scraps
- Scheme

Source:

- Book 1
- Page 178

GOAL

One company produces newsprint on rolls 1,5 meters wide and 70 meters long. The customers request for the next month follow below:

Order				Cutting Scheme				
Items	Rolls	Length	Width	1	2	3	4	5
1	70	70 m	80 cm	1	1	-	-	-
2	100	70 m	60 cm	1	-	2	1	-
3	120	70 m	50 cm	-	1		1	3
Useful Width			m	1.4	1.3	1.2	1.1	1.5
Scraps			cm	10	20	30	40	0

Elaborate the model of form to minimize the losses opting.

The following is the block of data for use in the model:

```
MODEL:
SETS:
 PRODUCT : SCRAPS, PRODUCE;
 RESOURCE: ORDER;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,        ORDER =
 R80CM            70
 R60CM            100
 R50CM            120;
! Products attributes;
 PRODUCT,  SCRAPS  =
 CS1              10
 CS2              20
 CS3              30
 CS4              40
 CS5              0;
! Required        CS1   CS2   CS3   CS4   CS5;
 USAGE   =        1     1     0     0     0     ! R80CM;
                  1     0     2     1     0     ! R60CM;
                  0     1     0     1     3;    ! R50CM;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM( PRODUCT( p): SCRAPS( p) * PRODUCE( p));
! The order constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) = ORDER( r);
        @GIN( PRODUCE););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  CUTTING SCHEME ( Rolls of 70m x 1.50 m ):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SCRAPS ( cm ):", @NEWLINE( 1));
@TABLE(SCRAPS);
@WRITE(" ", @NEWLINE( 1), "  ORDER ( Rolls: (70 m x 80, 60, 50 cm ):", @NEWLINE( 1));
@TABLE(ORDER);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
!Solution report;
@WRITE(" ", @NEWLINE( 1),"  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I) | PRODUCE(I) #GT# 0: '  Cutting Scheme:',
        @FORMAT(PRODUCT( I),'-3s'), ' was used: ',
        @FORMAT(PRODUCE(I),'%2.0f'), ' rollers x scraps p/roll: ',
        @FORMAT(SCRAPS(I),'%4.1f'), ' cm = total scraps:',
        @FORMAT(SCRAPS(I) * PRODUCE(I),'%6.1f'),' cm',
@NEWLINE( 1));
@WRITE("  TOTAL:", 16*' ',
        @FORMAT(@SUM(PRODUCT(I): PRODUCE(I)),'%9.0f'), ' rollers', 40*' ',
        @FORMAT(@SUM(PRODUCT(I): SCRAPS(I) * PRODUCE(I)),'%7.1f'),' cm',
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
CUTTING SCHEME ( Rolls of 70 m x 1.50 m ):
             CS1       CS2       CS3       CS4       CS5
 R80CM  1.000000  1.000000  0.000000  0.000000  0.000000
 R60CM  1.000000  0.000000  2.000000  1.000000  0.000000
 R50CM  0.000000  1.000000  0.000000  1.000000  3.000000

SCRAPS ( cm ):
 CS1    10.00000
 CS2    20.00000
 CS3    30.00000
 CS4    40.00000
 CS5    0.000000

ORDER ( Rolls: (70 m x 80,60,50 cm ):
 R80CM  70.00000
 R60CM  100.0000
 R50CM  120.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
 SOLUTION:
 Global optimal solution found.
 Objective value:                                                     1150.000
 Objective bound:                                                     1150.000
 Infeasibilities:                                                     0.000000

IDEAL CUTTING PROGRAM:
Cutting Scheme:CS1 was used: 70 rollers x scraps p/roll: 10.0 cm = total scraps: 700.0
Cutting Scheme:CS3 was used: 15 rollers x scraps p/roll: 30.0 cm = total scraps: 450.0
Cutting Scheme:CS5 was used: 40 rollers x scraps p/roll:  0.0 cm = total scraps:   0.0
TOTAL:                      125 rollers                                       1150.0
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Scraps
- Scheme
- Bar

Source:

- Book 1
- Page 107

GOAL

The company orders iron bars according to the order below and has 7 meters bars. To minimize losses, the cut-off scheme was developed. The information used in the model is described below:

Order			Cutting Scheme (bars 7 m)								
Bars	m	Total		1	2	3	4	5	6	7	8
50	2	100	un	2	0	1	1	1	0	3	0
60	3	180		0	1	0	0	1	2	0	0
90	4	360		0	0	0	1	0	0	0	1
Useful			m	4	3	2	6	5	6	6	4
Scraps			m	3	4	5	1	2	1	1	3

Develop the model in a way to minimize losses.

```
MODEL:
SETS:
 PRODUCT : SCRAPS, PRODUCE;
 RESOURCE: ORDER;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,        ORDER  =
 Item1_Bars_2m    50
 Item2_Bars_3m    60
 Item3_Bars_4m    90;
! Products attributes;
 PRODUCT,         SCRAPS  =
 CS1              3
 CS2              4
 CS3              5
 CS4              1
 CS5              2
 CS6              1
 CS7              1
 CS8              3;
! Required        CS1   CS2   CS3   CS4   CS5   CS6   CS7   CS8;
 USAGE      =     2     0     1     1     1     0     3     0     ! Item1_Bars_2m;
                  0     1     0     0     1     2     0     0     ! Item2_Bars_3m;
                  0     0     0     1     0     0     0     1;    ! Item3_Bars_4m;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN = @SUM( PRODUCT( p): SCRAPS( p) * PRODUCE( p));
! The order constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) = ORDER( r);
        @GIN( PRODUCE););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  CUTTING STOCK - Bar 6 m:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SCRAPS (m):", @NEWLINE( 1));
@TABLE(SCRAPS);
@WRITE(" ", @NEWLINE( 1), "  ORDER (Bars):", @NEWLINE( 1));
@TABLE(ORDER);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( I) | PRODUCE( I) #GT# 0: '  Cutting: ',
        @FORMAT(PRODUCT( I),'-3s'),', was used:',
        @FORMAT(PRODUCE( I),'%3.0f'), ' bars  x scraps p/bar:',
        @FORMAT(SCRAPS( I),'%2.0f'), ' m  =  Scrap total:',
        @FORMAT(SCRAPS( I) * PRODUCE(I),'%5.1f'),' m',
@NEWLINE( 1));
@WRITE("  TOTAL:", 11*' ',
        @FORMAT(@SUM(PRODUCT(I): PRODUCE(I)),'%9.0f'), ' bars', 35*' ',
        @FORMAT(@SUM(PRODUCT(I): SCRAPS(I) * PRODUCE(I)),'%8.1f'),' m',
@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
CUTTING STOCK - Bar 6 m:
                     CS1       CS2       CS3       CS4       CS5       CS6       CS7       CS8
 ITEM1_BARS_2M  2.000000  0.000000  1.000000  1.000000  1.000000  0.000000  3.000000  0.000000
 ITEM2_BARS_3M  0.000000  1.000000  0.000000  0.000000  1.000000  2.000000  0.000000  0.000000
 ITEM3_BARS_4M  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  1.000000

SCRAPS (m):
 CS1  3.000000
 CS2  4.000000
 CS3  5.000000
 CS4  1.000000
 CS5  2.000000
 CS6  1.000000
 CS7  1.000000
 CS8  3.000000

ORDER (Bars):
 ITEM1_BARS_2M  50.00000
 ITEM2_BARS_3M  60.00000
 ITEM3_BARS_4M  90.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                             200.0000
Objective bound:                                             200.0000
Infeasibilities:                                             0.000000

IDEAL CUTTING PROGRAM:
Cutting: CS4, was used: 50 bars  x scraps p/bar: 1 m  =  Scrap total: 50.0 m
Cutting: CS6, was used: 30 bars  x scraps p/bar: 1 m  =  Scrap total: 30.0 m
Cutting: CS8, was used: 40 bars  x scraps p/bar: 3 m  =  Scrap total:120.0 m
TOTAL:                 120 bars                                     200.0 m
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Scraps
- Scheme
- Plates

Source:

- Book 1
- Page 108

GOAL

A company has ordered to cut boards according to the order below, using plates with 6 meters x 3 meters. To minimize the leftovers, the cutting scheme was elaborated. The information used in the model is described below:

Order				Cutting Scheme Plates 18m2 (6 x 3)							
Plates	L	W	m2		1	2	3	4	5	6	7
70	3	3	630	un	2	1	0	0	0	1	0
60	4	3	720	un	0	0	1	0	0	0	1
80	2	2	320	un	0	1	1	3	1	0	0
Useful				m2	18	13	16	12	4	9	12
Scraps				m2	0	5	2	6	14	9	6

Elaborate the model in a way to minimize the leftovers by opting for the best cutting scheme.

```
MODEL:
SETS:
 PRODUCT : SCRAPS, PRODUCE;
 RESOURCE: ORDER;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,              ORDER =
 Item1_Plates_3m_x_3m   70
 Item2_Plates_4m_x_3m   60
 Item3_Plates_2m_x_2m   80;
! Products attributes;
 PRODUCT,               SCRAPS =
 CS1                    0
 CS2                    5
 CS3                    2
 CS4                    6
 CS5                    14
 CS6                    9
 CS7                    6;
! Required              CS1   CS2   CS3   CS4   CS5   CS6   CS7;
 USAGE   =              2     1     0     0     0     1     0      ! Item1_Plates_3m_x_3m;
                        0     0     1     0     0     0     1      ! Item2_Plates_4m_x_3m;
                        0     1     1     3     1     0     0;     ! Item3_Plates_2m_x_2m;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( PRODUCT( p): SCRAPS( p) * PRODUCE( p));
! The order constraints;
@FOR( RESOURCE( r):
      [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) = ORDER( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  CUTTING STOCK - Plates 6m x 3m:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SCRAPS (un):", @NEWLINE( 1));
@TABLE(SCRAPS);
@WRITE(" ", @NEWLINE( 1), "  ORDER (Plates):", @NEWLINE( 1));
@TABLE(ORDER);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I) | PRODUCE(I) #GT# 0: '  Cutting: ',
      @FORMAT(PRODUCT( I),'-3s'),', was used:',
      @FORMAT(PRODUCE( I),'%3.0f'), ' plates  x  scraps p/plates:',
      @FORMAT(SCRAPS( I),'%2.0f'), ' un  =  total scraps:',
      @FORMAT(SCRAPS( I) * PRODUCE( I),'%4.0f'),' un',
@NEWLINE( 1));
@WRITE("  TOTAL:", 11*' ',
      @FORMAT(@SUM(PRODUCT(I): PRODUCE(I)),'%9.0f'), ' plates', 40*' ',
      @FORMAT(@SUM(PRODUCT(I): SCRAPS(I) * PRODUCE(I)),'%8.0f'),' un',
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
CUTTING STOCK - Plates 6 m x 3 m:
                            CS1       CS2       CS3       CS4       CS5       CS6
 ITEM1_PLATES_3M_X_3M  2.000000  1.000000  0.000000  0.000000  0.000000  1.000000
 ITEM2_PLATES_4M_X_3M  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000
 ITEM3_PLATES_2M_X_2M  0.000000  1.000000  1.000000  3.000000  1.000000  0.000000

                            CS7
 ITEM1_PLATES_3M_X_3M  0.000000
 ITEM2_PLATES_4M_X_3M  1.000000
 ITEM3_PLATES_2M_X_2M  0.000000

SCRAPS (un):
 CS1  0.000000
 CS2  5.000000
 CS3  2.000000
 CS4  6.000000
 CS5  14.00000
 CS6  9.000000
 CS7  6.000000

ORDER (plates):
 ITEM1_PLATES_3M_X_3M  70.00000
 ITEM2_PLATES_4M_X_3M  60.00000
 ITEM3_PLATES_2M_X_2M  80.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                      160.0000
Objective bound:                                                      160.0000
Infeasibilities:                                                      0.000000


IDEAL CUTTING PROGRAM:
Cutting: CS1, was used: 35 plates  x  scraps p/plates: 0 un  =  total scraps:   0 un
Cutting: CS3, was used: 60 plates  x  scraps p/plates: 2 un  =  total scraps: 120 un
Cutting: CS4, was used:  7 plates  x  scraps p/plates: 6 un  =  total scraps:  40 un
TOTAL:                 102 plates                                          160 un
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Scraps
- Scheme
- Plates

Source:

- Book 2
- Page 243

GOAL

A distributor of stocked wood veneer stores in the standard length of 25 meters, which can be cut into lengths that vary according to the customer's need.

This distributor has just received an order of 5000 plates of 7 meters, 1200 plates of 9 meters and 300 plates of 11 meters.

The distributor's manager identified six ways to cut the 25 meters boards to meet this request. The six cuts are summarized in the cut stock.

One possibility would be to cut the 25 meters plate into three 7 meters pieces and not cut 9 meters and 11 meters. The information used in the model is described below:

Order			Cutting Scheme (plates 25 m)						
Plates	Length	Total		1	2	3	4	5	6
5000	7	35000	un	3	2	2	1	0	0
1200	9	10800	un	0	1	0	2	1	0
300	11	3300	un	0	0	1	0	1	2
Useful Length			m	21	23	25	25	20	22
Scraps			m	4	2	0	0	5	3

It is important to note that 21 m will be cut, and there will be a loss of 4 m of plywood. The manager wants to fulfill the request by using as few 25 m plates as possible.

```
MODEL:
SETS:
 PRODUCT : SCRAPS, PRODUCE;
 RESOURCE: ORDER;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,              ORDER  =
 Item1_Plates__7m_long  5000
 Item2_Plates__9m_long  1200
 Item3_Plates_11m_long  300;
! Products attributes;
 PRODUCT,               SCRAPS   =
 CS1                    4
 CS2                    2
 CS3                    0
 CS4                    0
 CS5                    5
 CS6                    3;
! Required              CS1   CS2   CS3   CS4   CS5   CS6;
 USAGE       =          3     2     2     1     0     0     ! Item1_Plates__7m_long;
                        0     1     0     2     1     0     ! Item2_Plates__9m_long;
                        0     0     1     0     1     2;    ! Item3_Plates_11m_long;
ENDDATA
SUBMODEL MIN4:
[OBJ] MIN = @SUM( PRODUCT( p): SCRAPS( p) * PRODUCE( p));
! The order constraints;
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) = ORDER( r);
        @GIN(PRODUCE));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  CUTTING STOCK - Plates 25m Long:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SCRAPS (m):", @NEWLINE( 1));
@TABLE(SCRAPS);
@WRITE(" ", @NEWLINE( 1), "  ORDER (Plates):", @NEWLINE( 1));
@TABLE(ORDER);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN4);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I) | PRODUCE(I) #GT# 0: '  Cutting: ',
        @FORMAT(PRODUCT( I),'-3s'),', was used:',
        @FORMAT(PRODUCE(I),'%5.0f'), ' plates  x  scraps p/plates:',
        @FORMAT(SCRAPS(I),'%2.0f'), 'm  =  total scraps:',
        @FORMAT(SCRAPS(I) * PRODUCE(I),'%5.0f'),' m',
@NEWLINE( 1));
@WRITE("  TOTAL:", 16*' ',
        @FORMAT(@SUM(PRODUCT(I): PRODUCE(I)),'%6.0f'), ' plates', 41*' ',
        @FORMAT(@SUM(PRODUCT(I): SCRAPS(I) * PRODUCE(I)),'%6.0f'),' m',
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
CUTTING STOCK - Plates 25m Long:
                            CS1       CS2       CS3       CS4       CS5       CS6
 ITEM1_PLATES__7M_LONG  3.000000  2.000000  2.000000  1.000000  0.000000  0.000000
 ITEM2_PLATES__9M_LONG  0.000000  1.000000  0.000000  2.000000  1.000000  0.000000
 ITEM3_PLATES_11M_LONG  0.000000  0.000000  1.000000  0.000000  1.000000  2.000000

SCRAPS (m):
 CS1  4.000000
 CS2  2.000000
 CS3  0.000000
 CS4  0.000000
 CS5  5.000000
 CS6  3.000000

ORDER (Plates):
 ITEM1_PLATES__7M_LONG  5000.000
 ITEM2_PLATES__9M_LONG  1200.000
 ITEM3_PLATES_11M_LONG  300.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                      5075.000
Objective bound:                                                      5075.000
Infeasibilities:                                                      0.000000

 IDEAL CUTTING PROGRAM:
 Cutting: CS1, was used:  668 plates  x  scraps p/plates: 4m  =  total scraps: 2672 m
 Cutting: CS2, was used: 1200 plates  x  scraps p/plates: 2m  =  total scraps: 2400 m
 Cutting: CS3, was used:  298 plates  x  scraps p/plates: 0m  =  total scraps:    0 m
 Cutting: CS6, was used:    1 plates  x  scraps p/plates: 3m  =  total scraps:    3 m
 TOTAL:                  2167 plates                                          5075 m
```

5

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Scraps
- Scheme
- Carpet
- Rolls

Source:
- Book 2
- Page 244

GOAL

Carpe International has received an application to provide carpets for a new office building, as per the quantities below:

		Type			T1			T2		
Rolls		Size		m	1000 x 14			1000 x 18		
		Cost		$	1,000.00			1,400.00		
Order					Cutting Scheme					
Item	Rolls	Length	Width		1	2	3	4	5	6
1	40	4000	4	m	3	1	0	1	0	1
2	200	20000	9	m	0	1	0	1	1	0
3	90	9000	12	m	0	0	1	0	0	1
Useful Width				m	12	13	12	17	18	16
Scraps				m	2	1	2	1	0	2

The company can purchase two types of carpet rolls, which must be cut to fit the order. One of the types has:

- 14 meters wide by 100 meters long and costs $1000 per roll; and the other
- 18 meters wide by 100 meters long and costs $1400 per roll.

The manager wants to determine how many rolls each type should buy and how they should be cut so as to have the lowest possible cost.

```
MODEL:
SETS:
 PRODUCT : SCRAPS, COST,  PRODUCE;
 RESOURCE: ORDER;
 RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,             ORDER       =
 Item1_4000m_x__4m     40
 Item2_20000m_x__9m    200
 Item3_9000m_x_12m     90;
! Products attributes;
 PRODUCT,              SCRAPS,     COST  =
 CS1                   2           1000
 CS2                   1           1000
 CS3                   2           1000
 CS4                   1           1400
 CS5                   0           1400
 CS6                   2           1400;
! Required             CS1   CS2   CS3   CS4   CS5   CS6;
 USAGE     =           3     1     0     0     1     2      ! Item1_4000 m_x__4 m;
                       0     1     0     2     0     1      ! Item2_20000 m_x__9 m;
                       0     0     1     0     1     0;     ! Item3_9000 m_x_12 m;
ENDDATA
SUBMODEL MIN5:
[OBJ] MIN = @SUM( PRODUCT( p): SCRAPS( p) * PRODUCE( p));
@FOR( RESOURCE( r):
        [CON] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) = ORDER( r); );
ENDSUBMODEL
CALC:
@SET('TERSEO',1);    ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);    ! Post status windows, 1 Yes, 0 No;
! Data block;
@WRITE("  CUTTING SCHEME: ", @NEWLINE( 1));
@WRITE("   CS1 .. CS3 (Rolls: 100 m x 14 m)",  "       " , "   CS4 .. CS6 (Rolls: 100 m x 18 m)", @NEWLINE( 2));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  COST per rolls:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SCRAPS ( m ):", @NEWLINE( 1));
@TABLE(SCRAPS);
@WRITE(" ", @NEWLINE( 1), "  ORDER ( Rolls ):", @NEWLINE( 1));
@TABLE(ORDER);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I) | PRODUCE(I) #GT# 0: '  Cutting Scheme:',
        @FORMAT(PRODUCT( I),'-3s'),', was used:',
        @FORMAT(PRODUCE(I), '%3.0f'),' rollers x ', 'scraps p/roll:',
        @FORMAT(SCRAPS(I),'%3.0f'), ' m  = Total ',
        @FORMAT(SCRAPS(I) * PRODUCE(I),'%3.0f'),' m',
@NEWLINE( 1));
@WRITE('  TOTAL SCRAPS:', 53*' ',@FORMAT(@SUM(PRODUCT(I):SCRAPS(I) * PRODUCE(I)),'%9.0f'),' m',@NEWLINE(2));
@WRITEFOR( PRODUCT(I) | PRODUCE(I) #GT# 0: '  Cutting Scheme:',
        @FORMAT(PRODUCT( I),'-3s'),', was used:',
        @FORMAT(PRODUCE(I), '%3.0f'),' rollers x Cost: $',
        @FORMAT(COST(I),'%7.2f'), ' = Total: $',
        @FORMAT(COST(I) * PRODUCE(I),'%9.2f'),
@NEWLINE( 1));
@WRITE('  TOTAL COST:', 56*' ','$', @FORMAT(@SUM(PRODUCT(I):COST(I) * PRODUCE(I)),'%9.2f'),@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
CUTTING SCHEME:
 CS1 .. CS3 (Rolls: 100 m x 14 m)
 CS4 .. CS6 (Rolls: 100 m x 18 m)

                          CS1       CS2       CS3       CS4       CS5       CS6
 ITEM1_4000M_X__4M   3.000000  1.000000  0.000000  0.000000  1.000000  2.000000
 ITEM2_20000M_X__9M  0.000000  1.000000  0.000000  2.000000  0.000000  1.000000
 ITEM3_9000M_X_12M   0.000000  0.000000  1.000000  0.000000  1.000000  0.000000

COST per rolls:
 CS1  1000.000
 CS2  1000.000
 CS3  1000.000
 CS4  1400.000
 CS5  1400.000
 CS6  1400.000

SCRAPS ( m ):
 CS1  2.000000
 CS2  1.000000
 CS3  2.000000
 CS4  1.000000
 CS5  0.000000
 CS6  2.000000

ORDER ( Rolls ):
 ITEM1_4000M_X__4M   40.00000
 ITEM2_20000M_X__9M  200.0000
 ITEM3_9000M_X_12M   90.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                   200.0000
Objective bound:                                                   200.0000
Infeasibilities:                                                   0.000000

IDEAL CUTTING PROGRAM:
Cutting Scheme:CS3, was used: 50 rollers x scraps p/roll:  2 m  = Total 100 m
Cutting Scheme:CS4, was used:100 rollers x scraps p/roll:  1 m  = Total 100 m
Cutting Scheme:CS5, was used: 40 rollers x scraps p/roll:  0 m  = Total   0 m
TOTAL SCRAPS:                                                          200 m

Cutting Scheme:CS3, was used: 50 rollers x Cost: $1000.00 = Total: $ 50000.00
Cutting Scheme:CS4, was used:100 rollers x Cost: $1400.00 = Total: $140000.00
Cutting Scheme:CS5, was used: 40 rollers x Cost: $1400.00 = Total: $ 56000.00
TOTAL COST:                                                        $246000.00
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Scraps
- Scheme
- Waste

Source:

- Book 5
- Page 119

GOAL

COOLDOT Appliance Company produces a wide range of large household appliances such as refrigerators and stoves. A significant portion of the raw material cost is due to the purchase of sheet steel.

Currently, sheet steel is purchased in coils in three different widths: 72 inches, 48 inches, and 36 inches.

In the manufacturing process, eight different widths of sheet steel are required: 60, 56, 42, 38, 34, 24, 15, and 10 inches.

All uses require the same quality and thickness of steel. A continuing problem is trim waste. For example, one way of cutting a 72-inch width coil is to slit it into one 38-inch width coil and two 15-inch width coils.

There will then be a 4-inch coil of trim waste that must be scrapped.

The cost per linear foot of the three different raw material widths are 15 cents for the 36-inch width, 19 cents for the 48-inch width, and 28 cents for the 72-inch width.

Simple arithmetic reveals the costs per inch x foot of the three widths are 15/36 = 0.416667 cents/(inch x foot), 0.395833 cents/(inch x foot), and 0.388889 cents/(inch x foot) for the 36, 48, and 72-inch widths, respectively.

The coils may be slit in any feasible solution. The possible cutting patterns for efficiently slitting the three raw material widths are tabulated below.

For example, pattern C_4 corresponds to cutting a 72-inch width coil into one 24-inch width and four 10-inch widths with 8-inch left over as trim waste.

The lengths of the various widths required in this planning period are:

The raw material availabilities this planning period are 1600 Ft. of the 72-inch coils and 10000 Ft. each of the 48-inch and 36-inch widths.

How many feet of each pattern should be cut to minimize cost while satisfying the requirements of the various widths?

Can you predict beforehand the amount of 36-inch material used?

```
MODEL:
SETS:
! Each raw material has a: RWDTH=Raw material width, T=Total used, W=Waste total, C=Cost per unit,
WCOST=Waste cost, S=      Supply available;
  RM: RWDTH,T, W, C, WCOST, S;
! Each Finished good has a: FWDTH Width,   REQ = units Required,  X=Xtra produced;
  FG: FWDTH, REQ, X;
  PATTERN: USERM, WASTE, AMT;
  PXF( PATTERN, FG): NUM;
ENDSETS
DATA:
! Finished good attributes;
  FG,         FWDTH,      REQ    =
  F60         60          500
  F56         56          400
  F42         42          300
  F38         38          450
  F34         34          350
  F24         24          100
  F15         15          800
  F10         10          1000;
! Raw material attributes;
  RM,         RWDTH,      C,     WCOST,       S       =
  R72         72          0.28   0.00388889   1600
  R48         48          0.19   0.00395833   10000
  R36         36          0.15   0.00416667   10000;
! Index of R.M. that each pattern uses;
USERM       = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3;
! How many of each F.G. are in each R.M. pattern;
!................................ Number to Cut of the Required Width............................ Pattern;
!           60"    56"    42"    38"    34"    24"    15"    10";
NUM=        1      0      0      0      0      0      0      1             !A1-72";
            0      1      0      0      0      0      1      0             !A2-72";
            0      1      0      0      0      0      0      1             !A3-72";
            0      0      1      0      0      1      0      0             !A4-72";
            0      0      1      0      0      0      2      0             !A5-72";
            0      0      1      0      0      0      1      1             !A6-72";
            0      0      1      0      0      0      0      3             !A7-72";
            0      0      0      1      1      0      0      0             !A8-72";
            0      0      0      1      0      1      0      1             !A9-72";
            0      0      0      1      0      0      2      0             !B0-72";
            0      0      0      1      0      0      1      1             !B1-72";
            0      0      0      1      0      0      0      3             !B2-72";
            0      0      0      0      2      0      0      0             !B3-72";
            0      0      0      0      1      1      0      1             !B4-72";
            0      0      0      0      1      0      2      0             !B5-72";
            0      0      0      0      1      0      1      2             !B6-72";
            0      0      0      0      1      0      0      3             !B7-72";
            0      0      0      0      0      3      0      0             !B8-72";
            0      0      0      0      0      2      1      0             !B9-72";
            0      0      0      0      0      2      0      2             !C0-72";
            0      0      0      0      0      1      3      0             !C1-72";
            0      0      0      0      0      1      2      1             !C2-72";
            0      0      0      0      0      1      1      3             !C3-72";
            0      0      0      0      0      1      0      4             !C4-72";
            0      0      0      0      0      0      4      1             !C5-72";
            0      0      0      0      0      0      3      2             !C6-72";
            0      0      0      0      0      0      2      4             !C7-72";
            0      0      0      0      0      0      1      5             !C8-72";
            0      0      0      0      0      0      0      7             !C9-72";
            0      0      1      0      0      0      0      0             !D0-48";
            0      0      0      1      0      0      0      1             !D1-48";
            0      0      0      0      1      0      0      1             !D2-48";
            0      0      0      0      0      2      0      0             !D3-48";
            0      0      0      0      0      1      1      0             !D4-48";
            0      0      0      0      0      1      0      2             !D5-48";
            0      0      0      0      0      0      3      0             !D6-48";
            0      0      0      0      0      0      2      1             !D7-48";
            0      0      0      0      0      0      1      3             !D8-48";
            0      0      0      0      0      0      0      4             !D9-48";
            0      0      0      0      1      0      0      0             !E0-36";
            0      0      0      0      0      1      0      1             !E1-36";
            0      0      0      0      0      0      2      0             !E2-36";
            0      0      0      0      0      0      1      2             !E3-36";
            0      0      0      0      0      0      0      3;            !E4-36";

ENDDATA
```

```
SUBMODEL MIN6:
[OBJ] MIN = TPROFIT;
 TPROFIT = @SUM(RM(I): C(I)*T(I) );
! Compute total cost of waste;
 TOTWASTE = @SUM( RM(I): WCOST(I)*W(I) );
@FOR( RM( I):
        T( I) = @SUM( PATTERN( K)| USERM(K) #EQ# I: AMT( K));
        ! Raw material supply constraints;
        T(I) <= S(I););
! Must produce at least amount required of each F.G.;
@FOR( FG(J):
        @SUM(PATTERN(K): NUM(K,J)*AMT(K)) = REQ(J) + X(J););
 ! Turn this on to get integer solutions;
 @FOR( PATTERN(K): @GIN(AMT(K)));
! Waste related computations;
!  Compute waste associated with each pattern;
 @FOR( PATTERN(K):
        WASTE(K) = RWDTH(USERM(K)) - @SUM(FG(J): FWDTH(J)*NUM(K,J)););
! Waste for each R.M. in this solution;
 @FOR( RM( I):
         W(I) = @SUM( PATTERN( K)| USERM(K) #EQ# I: WASTE(K)*AMT( K)););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Fixed line length
@SET('LINLEN',120);
! Precision in digits for standard solution reports;
@SET('PRECIS',6);
! Data block;
@WRITE(" DATA:", @NEWLINE( 1));
@WRITE("  CUTTING SCHEME:", @NEWLINE( 1));
@TABLE(NUM);
@WRITE(" ", @NEWLINE( 1), "  WHIDTH (inches ):", @NEWLINE( 1));
@TABLE(FWDTH);
@WRITE(" ", @NEWLINE( 1), "  PROFIT ( Waste ):", @NEWLINE( 1));
@TABLE(C);
@WRITE(" ", @NEWLINE( 1), " AVAILABLE PROVIDER( Waste ):", @NEWLINE( 1));
@TABLE(S);
@WRITE(" ", @NEWLINE( 1), " REQUIRED ( feet ):", @NEWLINE( 1));
@TABLE(REQ);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN6);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL CUTTING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RM(J): '  Raw Width:',
        @FORMAT(RWDTH(J),'%3.0f'),' inches, used: ',
        @FORMAT(T(J),'%4.0f'),' waste x unity cost: $',
        @FORMAT(C(J),'%4.2f'), ' = Total: $',
        @FORMAT(T(J) * C(J),'%7.3f'),
@NEWLINE( 1));
@WRITE("  TOTAL:", 1*' ', @FORMAT(@SUM(RM(J): RWDTH(J)),'%6.0f'), ' inches', 8*' ',
        @FORMAT(@SUM(RM(J): T(J)),'%4.0f'), ' waste',30*' ','$',
        @FORMAT(@SUM(RM(J): T(J) * C(J)),'%7.3f'),
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

DATA:
CUTTING SCHEME:

	F60	F56	F42	F38	F34	F24	F15	F10
1	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000
2	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000
3	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000
4	0.00000	0.00000	1.00000	0.00000	0.00000	1.00000	0.00000	0.00000
5	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	2.00000	0.00000
6	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	1.00000
7	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	3.00000
8	0.00000	0.00000	0.00000	1.00000	1.00000	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
10	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	2.00000	0.00000
11	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	1.00000	1.00000
12	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	3.00000
13	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000	0.00000	0.00000
14	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000	0.00000	1.00000
15	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	2.00000	0.00000
16	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	1.00000	2.00000
17	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	3.00000
18	0.00000	0.00000	0.00000	0.00000	0.00000	3.00000	0.00000	0.00000
19	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	1.00000	0.00000
20	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000	2.00000
21	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	3.00000	0.00000
22	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	2.00000	1.00000
23	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000	3.00000
24	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	4.00000
25	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	4.00000	1.00000
26	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.00000	2.00000
27	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	4.00000
28	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	5.00000
29	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	7.00000
30	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
31	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000
32	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	1.00000
33	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000	0.00000
34	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000	0.00000
35	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	2.00000
36	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.00000	0.00000
37	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	1.00000
38	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	3.00000
39	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	4.00000
40	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000
41	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	1.00000
42	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000
43	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	2.00000
44	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.00000

WHIDTH (inches):

F60	60.0000
F56	56.0000
F42	42.0000
F38	38.0000
F34	34.0000
F24	24.0000
F15	15.0000
F10	10.0000

COST (Waste):

R72	0.280000
R48	0.190000
R36	0.150000

AVAILABLE PROVIDER(Waste):

R72	1600.00
R48	10000.0
R36	10000.0

REQUIRED (feet):

F60	500.000
F56	400.000
F42	300.000
F38	450.000
F34	350.000
F24	100.000
F15	800.000
F10	1000.00

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                              466.340
Objective bound:                                              466.340
Infeasibilities:                                              0.00000

Model Class:MILP

IDEAL CUTTING PROGRAM:
Raw Width: 72 inches, used: 1599 waste x unity cost: $0.28 = Total: $447.720
Raw Width: 48 inches, used:   98 waste x unity cost: $0.19 = Total: $ 18.620
Raw Width: 36 inches, used:    0 waste x unity cost: $0.15 = Total: $  0.000
TOTAL:     156 inches        1697 waste                               $466.340
```

BLOCK 12

Block: METALLURGY

How to fulfill an order for an aluminum alloy, for example, considering the availability of different types of ore in stock and the technical needs of the composition respecting limits, in order to maximize profits?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 1
- Page 72

GOAL

A mining company wants to fulfill a contract of supply of 4 million tons per year of the Sinter Feed ore and, for that, it has the following minerals:

- Mineral A: Profit of $0.03 per kg - Stock of 600 kg;
- Mineral B: Profit of $0.05 per kg - Stock of 800 kg;
- Alloy Y: Alloy obtained by the selling profit of $0.08 per kg

Resources / Products		Alloy	Mineral A	Mineral B
Silicon	%	13	15	10
Iron	%	10	13	5
Aluminum	%	80	72	85
Iron Ore Stock	kg	0	600	800
Profit (p/k)	$	0.08	0.03	0.05

Regarding the aluminum alloy to be produced, it must meet the technical specifications that limit the amount of explicit chemical elements in the alloy column (Minimum of the Silicon and Iron, and Maximum of the Aluminum).

```
MODEL:
SETS:
 RESOURCE:;
 PRODUCT: PROFIT, STOCK, PRODUCE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE    =
 SILICON
 IRON
 ALUMINUM;
! Product attributes;
 PRODUCT,          PROFIT,      STOCK     =
 ALLOY             0.08         0
 MIN_A             0.03         600
 MIN_B             0.05         800;
! Required         ALLOY        MIN_A        MIN_B;
 USAGE     =       13           15           10           ! SILICON;
                   10           13           5            ! IRON;
                   80           72           85;          ! ALUMINUM;
ENDDATA
SUBMODEL MAX1:
[OBJ] MAX =  PROFIT(1) * PRODUCE(1) - PROFIT(2) * PRODUCE(2) - PROFIT(3) * PRODUCE(3);
! Equation  of profit;
PRODUCE(1) - PRODUCE(2) - PRODUCE(3) = 0;
! Restriction of Silicon use;
USAGE(1,1)/100  * PRODUCE(1) - USAGE(1,2)/100 * PRODUCE(2) - USAGE(1,3)/100 * PRODUCE (3) >=0;
! Restriction of Iron use;
USAGE(2,1)/100  * PRODUCE(1) - USAGE(2,2)/100 * PRODUCE(2) - USAGE(2,3)/100 * PRODUCE (3) >=0;
! Restriction of Aluminum use;
USAGE(3,1)/100  * PRODUCE(1) - USAGE(3,2)/100 * PRODUCE(2) - USAGE(3,3)/100 * PRODUCE (3) <= 0;
! Iron ore stock A;
PRODUCE(2) <= STOCK(2);
! Iron ore stock B;
PRODUCE(3) <= STOCK(3);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
        @FORMAT(PRODUCT(J),'-3s'), '  Profit p/kg: $',
        @FORMAT(PROFIT(J),'%4.2f'),' x ', @IF(J #EQ# 1, '+Alloy obtained:','-Used stock    :'), ' ' ,
        @FORMAT(PRODUCE(J), '%4.0f'),'kg  Total: $',
        @FORMAT(PROFIT(J) * PRODUCE(J),'%3.0f'), @IF(J #EQ# 1, '+','-'),
@NEWLINE( 1));
@WRITE('  Total:',39*' ',
        @FORMAT(-1*(PRODUCE(1)-PRODUCE(2)-PRODUCE(3)),'%4.0f'),'kg',9*' ', '$',
        @FORMAT(OBJ,'%3.0f'),
@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
              ALLOY     MIN_A     MIN_B
 SILICON   13.00000  15.00000  10.00000
 IRON      10.00000  13.00000  5.000000
 ALUMINUM  80.00000  72.00000  85.00000

PROFIT:
 ALLOY     0.080000
 MIN_A     0.030000
 MIN_B     0.050000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    49.00000
Infeasibilities:                                    0.000000

 IDEAL PLANNING PROGRAM:
 ALLOY  Profit p/kg: $0.08 x +Alloy obtained: 1300kg  Total: $104+
 MIN_A  Profit p/kg: $0.03 x -Used stock    :  500kg  Total: $ 15-
 MIN_B  Profit p/kg: $0.05 x -Used stock    :  800kg  Total: $ 40-
 Total:                                         0kg         $ 49
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 1
- Page 62

GOAL

A metallurgist wants to produce 2000 kg of aluminum alloy at minimal cost by mixing various ores. The alloy must meet engineering requirements that specify the maximum and minimum of various chemical elements that make it up.

The minimum and maximum requirements for each material are as follows:

- Minimum limit: MAT3 = 400 kg and MAT4=100 kg ;
- Maximum limit: MAT1 = 200 kg , MAT2=750 kg, MAT3=800 kg, MAT4=700 kg, MAT5=1500 kg;

RHS

- Minimum composition: AL=1500 kg and IS=250;
- Maximum composition: FE= 60 kg, CU=100 kg, MN=40 kg, MG=30 kg, SI=300 kg;

The MIN Limit line refers to the quantity that is desired to be forced into this process (for some reason, such as release of space, it is desired to force MAT3 and MAT4 raw materials in).

The ore cost and the composition of each of the chemical elements of the alloy are shown

Resources / Products			Mat 1	Mat 2	Mat 3	Mat 4	Mat 5	Al-Pure	Si-Pure	Min (kg)	Max (kg)
Composition	FE	%	0.15	0.04	0.02	0.04	0.02	0.01	0.03	-	60
	CU	%	0.03	0.05	0.08	0.02	0.06	0.01	-	-	100
	MN	%	0.02	0.04	0.01	0.02	0.02	-	-	-	40
	MG	%	0.02	0.03	-	-	0.01	-	-	-	30
	AL	%	0.70	0.75	0.80	0.75	0.80	0.97	-	1500	-
	SI	%	0.02	0.06	0.08	0.12	0.02	0.01	0.97	250	300
Limit	Min	kg	-	-	400	100	-	-	-	-	-
	Max	kg	200	750	800	700	1500	Infinite	Infinite	-	-
Cost		$	0.03	0.08	0.17	0.12	0.15	0.21	0.38	-	-

```
MODEL:
SETS:
 RESOURCE:RHS_MIN, RHS_MAX;
 PRODUCT: COST, LIMIT_MIN, LIMIT_MAX, PRODUCE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE ,          RHS_MIN,      RHS_MAX =
 FE                  1             60
 CU                  1             100
 MN                  1             40
 MG                  1             30
 AL                  1500           0
 SI                  250           300;
! Products attributes;
 PRODUCT,            COST,         LIMIT_MIN,    LIMIT_MAX =
 MAT1                0.03          0             200
 MAT2                0.08          0             750
 MAT3                0.17          400           800
 MAT4                0.12          100           700
 MAT5                0.15          0             1500
 AL_PURE             0.21          0             0
 SI_PURE             0.38          0             0;
! Required           MAT1          MAT2          MAT3          MAT4          MAT5          AL_PURE       SI_PURE;
 USAGE     =         0.15          0.04          0.02          0.04          0.02          0.01          0.03          !FE;
                     0.03          0.05          0.08          0.02          0.06          0.01          0.00          !CU;
                     0.02          0.04          0.01          0.02          0.02          0.00          0.00          !MN;
                     0.02          0.03          0.00          0.00          0.01          0.00          0.00          !MG;
                     0.70          0.75          0.80          0.75          0.80          0.97          0.00          !AL;
                     0.02          0.06          0.08          0.12          0.02          0.01          0.97;         !SI;
ENDDATA
SUBMODEL MIN2:
[OBJ] MIN =    @SUM(PRODUCT(I): COST(I) * PRODUCE(I));
! Limit;
[LIM]          @SUM(PRODUCT(I): PRODUCE(I))  = 2000;
! Maximum of iron limit;
[MAX_FE]       @SUM(PRODUCT(I): USAGE(1,I)  * PRODUCE(I)) <= 60 ;
! Maximum copper (CU) limit;
[MAX_CU]       @SUM(PRODUCT(I): USAGE(2,I)  * PRODUCE(I)) <= 100 ;
! Maximum of manganese limit;
[MAX_MN]       @SUM(PRODUCT(I): USAGE(3,I)  * PRODUCE(I)) <= 40 ;
! Maximum of magnesium limit;
[MAX_MG]       @SUM(PRODUCT(I): USAGE(4,I)  * PRODUCE(I)) <= 30 ;
! Minimum aluminum limit;
[MIN_AL]       @SUM(PRODUCT(I): USAGE(5,I)  * PRODUCE(I)) >= 1500 ;
! Maximum silicon limit;
[MAX_SI]       @SUM(PRODUCT(I): USAGE(6,I)  * PRODUCE(I)) <= 300 ;
! Minimum silicon limit;
[MIN_SI]       @SUM(PRODUCT(I): USAGE(6,I)  * PRODUCE(I)) >= 250 ;
! Maximum Mat1, Mat2, Mat3, Mat4 and Mat5;
@FOR(PRODUCT(I) | I #LE# 5: [MAX_MAT]  PRODUCE(I) <= LIMIT_MAX(I));
! Minimum Mat3 and Mat4;
 @FOR(PRODUCT(I) |  I #LT# 5: [MIN_MAT]  PRODUCE(I) >= LIMIT_MIN(I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
          @FORMAT(PRODUCT(J),'-7s'), '  Produce:',
          @FORMAT(PRODUCE(J),'%4.0f'),'kg x Unit cost: $',
          @FORMAT(COST(J), '%4.2f'),'  =  Total: $',
          @FORMAT(COST(J) * PRODUCE(J),'%6.2f'),
@NEWLINE( 1));
@WRITE('  Total:',11*' ', @FORMAT(PRODUCE(1)+PRODUCE(2)+PRODUCE(3)+PRODUCE(4)+
                                  PRODUCE(5)+ PRODUCE(6)+PRODUCE(7),'%4.0f'),'kg',30*' ', ' $',
                        @FORMAT(OBJ,'%5.2f'),@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
              MAT1       MAT2       MAT3       MAT4       MAT5    AL_PURE    SI_PURE
 FE      0.1500000  0.0400000  0.0200000  0.0400000  0.0200000  0.0100000  0.0300000
 CU      0.0300000  0.0500000  0.0800000  0.0200000  0.0600000  0.0100000  0.0000000
 MN      0.0200000  0.0400000  0.0100000  0.0200000  0.0200000  0.0000000  0.0000000
 MG      0.0200000  0.0300000  0.0000000  0.0000000  0.0100000  0.0000000  0.0000000
 AL      0.7000000  0.7500000  0.8000000  0.7500000  0.8000000  0.9700000  0.0000000
 SI      0.0200000  0.0600000  0.8000000  0.1200000  0.0200000  0.0100000  0.9700000

COST:
 MAT1         0.0300000
 MAT2         0.0800000
 MAT3         0.1700000
 MAT4         0.1200000
 MAT5         0.1500000
 AL_PURE      0.2100000
 SI_PURE      0.3800000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                      296.2166
Infeasibilities:                                      0.000000

IDEAL PLANNING PROGRAM:
MAT2     Produce: 665kg x Unit cost: $0.08  =  Total: $ 53.23
MAT3     Produce: 490kg x Unit cost: $0.17  =  Total: $ 83.34
MAT4     Produce: 424kg x Unit cost: $0.12  =  Total: $ 50.90
AL_PURE  Produce: 300kg x Unit cost: $0.21  =  Total: $ 62.92
SI_PURE  Produce: 121kg x Unit cost: $0.38  =  Total: $ 45.82
Total:           2000kg                               $296.22
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 11
- Ex.: 3

GOAL

A special alloy consisting of iron, coal, silicon and nickel can be obtained by mixing these pure minerals in addition to 2 types of recovered materials:

Recovered Material 1 (RM1)

Composition: iron - 60%; coal - 20%; silicon - 20%. Cost per kg: R $ 0.20.

Recovered Material 2 (RM2)

Composition: iron - 70%; coal: 20%; silicon - 5%; nickel - 5%. Cost per kg: $ 0.25.

The alloy should have the following final composition:

Resources / Products			RM1	RM2	Iron	Coal	Silicon	Nickel
Formula	RM1	%	-	-	60	20	20	-
	RM2	%	-	-	70	20	20	5
	Min	%	-	-	60	15	15	5
	Max	%	-	-	65	20	20	8
Cost p/kg		$	0.20	0.25	0.30	0.20	0.28	0.50

What should be the composition of the mixture in terms of available materials, with lower cost per kg?

```
MODEL:
SETS:
 RESOURCE:;
 PRODUCT: COST, PRODUCE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE   =
  MIN_IRON
  MAX_IRON
  MIN_COAL
  MAX_COAL
  MIN_SILICON
  MAX_SILICON
  MIN_NICKEL
  MAX_NICKEL
  RHS_MIN
  RHS_MAX;
! Product attributes;
  PRODUCT,          COST    =
  RM1               0.20
  RM2               0.25
  IRON              0.30
  COAL              0.20
  SILICON           0.28
  NICKEL            0.50;
! Required          RM1   RM2   IRON  COAL  SILICON     NICKEL;
  USAGE   =         60    70    1     0     0           0            ! MIN_IRON;
                    60    70    1     0     0           0            ! MAX_IRON;
                    20    20    0     1     0           0            ! MIN_COAL;
                    20    20    0     1     0           0            ! MAX_COAL;
                    20    20    0     0     1           0            ! MIN_SILICON;
                    20    0     0     0     1           0            ! MAX_SILICON ;
                    5     5     0     0     0           1            ! MIN_NICKEL;
                    5     5     0     0     0           1            ! MAX_NICKEL ;
                    0     0     60    15    15          5            ! RHS_MIN;
                    0     0     65    20    20          8;           ! RHS_MAX;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN =  @SUM(PRODUCT(I): COST(I) * PRODUCE(I));
[RM1A]        @SUM(PRODUCT(I)| I #LE# 3: USAGE(1,I)/100  * PRODUCE(I)) >= 0.60 ;
! RM1(60%)+RM2(70%)+ Iron min 60%;
[RM1B] @SUM(PRODUCT(I)| I #LE# 3: USAGE(2,I)/100  * PRODUCE(I)) <= 0.65 ;
! RM1(60%)+RM2(70%)+ Iron max 65%;
[RM1C] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 4: USAGE(3,I)/100  * PRODUCE(I)) >= 0.15;
!RM1(20%) + RM2(20%) + MIN COAL (15%) ;
[RM1D] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 4: USAGE(4,I)/100  * PRODUCE(I)) <= 0.20 ;
!RM1(20%) + RM2(20%) + MAX COAL (20%) ;
[RM1E] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 5: USAGE(5,I)/100  * PRODUCE(I)) >= 0.15 ;
! RM1(20%) + RM2(20%) + MIN Silicon (15%) ;
[RM1F] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 5: USAGE(6,I)/100  * PRODUCE(I)) <= 0.20 ;
! RM1(20%) + RM2(20%) + MAX Silicon (20%);
[RM2A] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 6: USAGE(7,I)/100  * PRODUCE(I)) >= 0.05 ;
! RM2(5%) + MIN Nickel (5%) ;
[RM2B] @SUM(PRODUCT(I)| I #LE# 2 #OR# I #EQ# 6: USAGE(8,I)/100  * PRODUCE(I)) <= 0.08;
! RM2(5%) + MAX Nickel (8%);
[LIMIT]    @SUM(PRODUCT(I): PRODUCE(I))  = 1;
ENDSUBMODEL
CALC:
@SET('TERSEO',1);                                                    ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);                                                    ! Post status windows, 1 Yes, 0 No;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  COST:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN3);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  Use:',
        @FORMAT(PRODUCE(J),'%2.0f'),'kg of: ',
        @FORMAT(PRODUCT(J),'-4s'), 'meets the lowest cost chemical composition: $',
        @FORMAT(COST(J), '%4.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                RM1       RM2       IRON      COAL      SILICON   NICKEL
 MIN_IRON     60.00000  70.00000  1.000000  0.000000  0.000000  0.000000
 MAX_IRON     60.00000  70.00000  1.000000  0.000000  0.000000  0.000000
 MIN_COAL     20.00000  20.00000  0.000000  1.000000  0.000000  0.000000
 MAX_COAL     20.00000  20.00000  0.000000  1.000000  0.000000  0.000000
 MIN_SILICON  20.00000  20.00000  0.000000  0.000000  1.000000  0.000000
 MAX_SILICON  20.00000  20.00000  0.000000  0.000000  1.000000  0.000000
 MIN_NICKEL   5.000000  5.000000  0.000000  0.000000  0.000000  1.000000
 MAX_NICKEL   5.000000  5.000000  0.000000  0.000000  0.000000  1.000000
 RHS_MIN      0.000000  0.000000  60.00000  15.00000  15.00000  5.000000
 RHS_MAX      0.000000  0.000000  65.00000  20.00000  20.00000  8.000000

COST:
 RM1       0.2000000
 RM2       0.2500000
 IRON      0.3000000
 COAL      0.2000000
 SILICON   0.2800000
 NICKEL    0.5000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                 0.2000000
Infeasibilities:                                 0.0000000

IDEAL PLANNING PROGRAM:
Use: 1kg of: RM1 meets the lowest cost chemical composition: $0.20
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 1
- Page 86

GOAL

One company plans to produce an alloy composed of two minerals A, B and sold for $0.08 per kilo.

Resources / Periods				Jan	Feb	Mar	Apr	May	Jun
Demand	Min		kg	800	900	900	800	800	800
	Max		kg	1000	1100	1200	1100	1000	900
Purchase	Ore	A	$	0.3	0.31	0.33	0.32	0.31	0.3
		B	$	0.4	0.51	0.55	0.54	0.52	0.5
Maximum Stocking	Ore	A	kg	1000	1000	1000	1000	1000	600
		B	kg	1000	1000	1000	1000	1000	800
	Cost		$	0.10	0.10	0.10	0.10	0.10	0.10
Price (p/kg)			$	2.00	2.00	2.00	2.00	2.00	2.00

The storage capacity of each miner (A, B) is 1000 kilos per month. The ore purchased will only be available the following month.

No purchase was made prior to January.The cost of stocking the raw material is $0.001 per kilo per month.

The final product can’t be stored; The final stock in June should be: A = 600 and B = 800.

You do not want to make a mining purchase in the month of June. Initial inventories are sufficient for January production and were purchased at: A $0.030 and B $0.040 per kilo.

The goal is to define purchasing and manufacturing planning to maximize profit.

```
MODEL:
SETS:
! Specification attributes;
 SPEC:      XL, XUA, XUB;
! Products attributes;
 PRODUCT:
 COST,      ! Stock sots;
 PRICE,     ! Sale price;
 P_ORE_A,   ! Purchase ore A;
 P_ORE_B,   ! Purchase ore B;
 S_ORE_A,   ! Stock ore A;
 S_ORE_B,   ! Stock ore B;
 D_MIN,     ! Demand Minimum;
 D_MAX,     ! Demand Maximum;
 L,         ! Profit;
 UA,        ! ORE A used in the month ;
 UB,        ! ORE B used in the month ;
 EA,        ! ORE A stock at the end of the month;
 EB,        ! ORE B stock at the end of the month;
 CA,        ! ORE A purchased in the month;
 CB;        ! ORE B purchased in the month;
ENDSETS
DATA:
! Specification attributes (%);
 SPEC,      XL,    XUA,   XUB =
 FE         0.13   0.15   0.10
 SI         0.10   0.13   0.05
 AL         0.80   0.72   0.85;

! Products attributes;
 PRODUCT,   COST   PRICE,      P_ORE_A,   P_ORE_B,   S_ORE_A,   S_ORE_B,   D_MIN,     D_MAX  =
 JAN        0.001  0.08        0.030      0.050      1000       1000       800        1000
 FEB        0.001  0.08        0.031      0.051      1000       1000       900        1100
 MAR        0.001  0.08        0.033      0.055      1000       1000       900        1200
 APR        0.001  0.08        0.032      0.054      1000       1000       800        1100
 MAY        0.001  0.08        0.031      0.052      1000       1000       800        1100
 JUN        0.001  0.08        0.030      0.050      600        800        800        900;
ENDDATA
```

```
SUBMODEL MAX4:
 [OBJ] MAX = PRI - CPA - CPB - CSA - CSB;
! Price;
PRI = @SUM(PRODUCT(J): PRICE(J) * L(J));
! Cost purchase ORE A;
CPA = @SUM(PRODUCT(J):P_ORE_A(J) * CA(J));
! Cost purchase ORE B;
CPB = @SUM(PRODUCT(J): P_ORE_B(J) * CB(J));
! Cost stocking ORE A;
CSA = @SUM(PRODUCT(J): COST(J) * EA(J));
! Cost stocking ORE B;
CSB = @SUM(PRODUCT(J): COST(J) * EB(J));
! LEAGUE OF MASS BALANCE WITH ORE;
@FOR(PRODUCT(J): L(J) + LX + UA(J) - UB(J) = 0);
! MAXIMUM CAPACITY OF STORAGE ORE A;
@FOR(PRODUCT(J): [_EA] EA(J) <= S_ORE_A(J));
! MAXIMUM CAPACITY OF STORAGE ORE B;
@FOR(PRODUCT(J): [_EB] EB(J) <= S_ORE_B(J));
! ORE MASS BALANCE A (USE, STOCK AND PURCHASE) ;
[_UA1] UA(1) + UA1 + EA(1) = 600;
@FOR(PRODUCT(J) | J #GT# 1:[_UA] UA(J) + UA2 + EA(J-1) - EA(J) - CA(J-1) = 0);
! ORE MASS BALANCE B (USE, STOCK AND PURCHASE) ;
[_UB1] UB(1) + UB1 + EB(1) = 800;
@FOR(PRODUCT(J) | J #GT# 1:[_UB]  UB(J) + UB2 + EB(J-1) - EB(J) - CB(J-1) = 0);
! FINAL PRODUCT SALES DEMAND ;
@FOR(PRODUCT(J): [_DEMMAX] L(J) <= D_MAX(J));
@FOR(PRODUCT(J): [_DEMMIN] L(J)   >= D_MIN(J));
! LEAGUE SPECIFICATION RESTRICTIONS;
! SPECIFICATION TO FE;
@FOR(PRODUCT(J): [_SPC_FE] XL(1)*L(J) - XUA(1)*UA(J) - XUB(1)*UB(J) >= 0);
! SPECIFICATION TO SI;
@FOR(PRODUCT(J): [_SPC_SI] XL(2)*L(J) - XUA(2)*UA(J) - XUB(2)*UB(J) >= 0);
! SPECIFICATION TO AL;
@FOR(PRODUCT(J): [_SPC_AL] XL(3)*L(J) - XUA(3)*UA(J) - XUB(3)*UB(J) <= 0);
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  STOCK ORE A (kg):", @NEWLINE( 1));
@TABLE(S_ORE_A);
@WRITE(" ", @NEWLINE( 1), "  STOCK ORE B (kg):", @NEWLINE( 1));
@TABLE(S_ORE_B);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM DEMAND(kg):", @NEWLINE( 1));
@TABLE(D_MIN);
@WRITE(" ", @NEWLINE( 1), "  MAXIMUM DEMAND(kg):", @NEWLINE( 1));
@TABLE(D_MAX);
@WRITE(" ", @NEWLINE( 1), "  PURCHASE COST ORE A (kg):", @NEWLINE( 1));
@TABLE(P_ORE_A);
@WRITE(" ", @NEWLINE( 1), "  PURCHASE COST ORE B (kg):", @NEWLINE( 1));
@TABLE(P_ORE_B);
@WRITE(" ", @NEWLINE( 1), " STOCK COST (P/kg) $:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SALE PRICE (P/kg) $:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX4);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITE('   Month:', 43*' ',' ALL', @NEWLINE(1));
@WRITE('   Produce: ', 39*' ',
        @FORMAT(@SUM(PRODUCT(J): D_MAX(J)), '%5.0f'),' kg',
@NEWLINE(1));
@WRITE(' + Revenue:', 37*' ', '$ ',@FORMAT(PRI,'%6.2f'),@NEWLINE(1));
@WRITE(' - Purchase cost:', 31*' ', '$ ',@FORMAT(CPA + CPB,'%6.2f'),@NEWLINE(1));
@WRITE(' - Stocking cost:', 30*' ',' $ ',@FORMAT(CSA + CSB,'%6.2f'),@NEWLINE(1));
@WRITE(' = Profit:', 37*' ',' $',
        @FORMAT(PRI - CPA - CPB - CSA - CSB,'%7.2f'),
@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
STOCK ORE A (kg):              PURCHASE COST ORE A (kg):
 JAN  1000.000                  JAN  0.030000
 FEB  1000.000                  FEB  0.031000
 MAR  1000.000                  MAR  0.033000
 APR  1000.000                  APR  0.032000
 MAY  1000.000                  MAY  0.031000
 JUN  600.0000                  JUN  0.030000

STOCK ORE B (kg):              PURCHASE COST ORE B (kg):
 JAN  1000.000                  JAN  0.050000
 FEB  1000.000                  FEB  0.051000
 MAR  1000.000                  MAR  0.055000
 APR  1000.000                  APR  0.054000
 MAY  1000.000                  MAY  0.052000
 JUN  800.0000                  JUN  0.050000

MINIMUM DEMAND(kg):           STOCK COST (P/kg) $:
 JAN  800.0000                  JAN  0.001000
 FEB  900.0000                  FEB  0.001000
 MAR  900.0000                  MAR  0.001000
 APR  800.0000                  APR  0.001000
 MAY  800.0000                  MAY  0.001000
 JUN  800.0000                  JUN  0.001000

MAXIMUM DEMAND(kg):            SALE PRICE (P/kg) $:
 JAN  1000.000                  JAN  0.080000
 FEB  1100.000                  FEB  0.080000
 MAR  1200.000                  MAR  0.080000
 APR  1100.000                  APR  0.080000
 MAY  1100.000                  MAY  0.080000
 JUN  900.0000                  JUN  0.080000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
 Global optimal solution found.
 Objective value:                              254.9000
 Infeasibilities:                              0.000000

 IDEAL PLANNING PROGRAM:
  Month:                                           ALL
  Produce:                                        6400 kg
+ Revenue:                                    $ 496.00
- Purchase cost:                              $ 239.30
- Stocking cost:                              $   1.80
= Profit:                                     $ 254.90
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 1
- Page 74

GOAL

One company wants to produce 3 aluminum alloys through two types of ore A, B, in which one wants to use the existing maximum stock to maximize profit from sales.

Resources / Products			Alloy 1			Alloy 2			Alloy 3		
Formula	SILICON	%	13	15	10	14	15	10	13	15	10
	IRON	%	10	13	5	11	13	5	8	13	5
	ALUMINUM	%	80	72	85	78	72	85	83	72	85
Stock		kg	-	600	800	-	600	800	-	600	800
Price (p/kg)		$	8.00	3.00	5.00	7.00	3.00	5.00	10.00	3.00	5.00

```
MODEL:

SETS:
 RESOURCE:;
 PRODUCT: PROFIT, STOCK, PRODUCE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS

DATA:
! Resources attributes;
 RESOURCE          =
 SILICON
 IRON
 ALUMINUM;
! Products attributes;
 PRODUCT ,         PROFIT,      STOCK       =
 ALLOY1_Y          8            0
 ALLOY1_A          3            600
 ALLOY1_B          5            800
 ALLOY2_Y          7            0
 ALLOY2_A          3            600
 ALLOY2_B          5            800
 ALLOY3_3          10           0
 ALLOY3_A          3            600
 ALLOY3_B          5            800;
! Required         A1_Y  A1_A  A1_B  A2_Y  A2_A  A2_B  AE_Y  A3_A  A3_B  ;
 USAGE      =      13    15    10    14    15    10    13    15    10    ! SILICON;
                   10    13    5     11    13    5     8     13    5     ! IRON;
                   80    72    85    78    72    85    83    72    85;   ! ALUMINUM;
ENDDATA

SUBMODEL MAX5:
[OBJ] MAX =  PROFIT(1)/100 * PRODUCE(1) + PROFIT(4)/100 * PRODUCE(4) + PROFIT(7)/100 * PRODUCE(7)  -
             PROFIT(2)/100 * PRODUCE(2)  - PROFIT(3)/100 * PRODUCE(3)  - PROFIT(5)/100 * PRODUCE(5)  -
             PROFIT(6)/100 * PRODUCE(6)  - PROFIT(8)/100 * PRODUCE(8)  - PROFIT(9)/100 * PRODUCE(9) ;
!Equalization of profit Allow 1 (Sales PROFIT - PROFIT PROFIT);
@SUM(PRODUCT(I)|I #LE# 3:  @IF( I #EQ# 1, PRODUCE(I),-1*PRODUCE(I)))  = 0;
!Silicon (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 13%);
USAGE(1,1)/100 * PRODUCE(1) - USAGE(1,2)/100 * PRODUCE(2) - USAGE(1,3)/100 * PRODUCE(3) > 0;
!Iron (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 10%);
USAGE(2,1)/100 * PRODUCE(1) - USAGE(2,2)/100 * PRODUCE(2) - USAGE(2,3)/100 * PRODUCE(3) > 0;
!Aluminum (Total weight of the A + B alloy divided by the total weight of Y must be greater than or equal to 80%);
USAGE(3,1)/100 * PRODUCE(1) - USAGE(3,2)/100 * PRODUCE(2) - USAGE(3,3)/100 * PRODUCE(3) < 0;
!Equalization of profit Allow 2 (Sales PROFIT - PROFIT PROFIT);
@SUM(PRODUCT(I)|I #EQ# 4 #OR# I #EQ# 5 #OR# I #EQ# 6: @IF( I #EQ# 4, PRODUCE(I),-1*PRODUCE(I)))  = 0;
!Silicon (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 14%;
USAGE(1,4)/100 * PRODUCE(4) - USAGE(1,5)/100 * PRODUCE(5) - USAGE(1,6)/100 * PRODUCE(6) > 0;
!Iron (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 11%);
USAGE(2,4)/100 * PRODUCE(4) - USAGE(2,5)/100 * PRODUCE(5) - USAGE(2,6)/100 * PRODUCE(6) > 0;
!Aluminum (Total weight of the A + B alloy divided by the total weight of Y must be greater than or equal to 78%);
USAGE(3,4)/100 * PRODUCE(4) - USAGE(3,5)/100 * PRODUCE(5) - USAGE(3,6)/100 * PRODUCE(6) < 0;
!Equalization of profit Allow 3 (Sales PROFIT - PROFIT PROFIT);
@SUM(PRODUCT(I)|I #EQ# 7 #OR# I #EQ# 8 #OR# I #EQ# 9: @IF( I #EQ# 7, PRODUCE(I),-1*PRODUCE(I)))  = 0;
!Silicon (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 13%;
USAGE(1,7)/100 * PRODUCE(7) - USAGE(1,8)/100 * PRODUCE(8) - USAGE(1,9)/100 * PRODUCE(9) > 0;
!Iron (Total weight of the A + B alloy divided by the total weight of Y must be less than or equal to 8%);
USAGE(2,7)/100 * PRODUCE(7) - USAGE(2,8)/100 * PRODUCE(8) - USAGE(2,9)/100 * PRODUCE(9) > 0;
!Aluminum (Total weight of the A + B alloy divided by the total weight of Y must be greater than or equal to 72%);
USAGE(3,7)/100 * PRODUCE(7) - USAGE(3,8)/100 * PRODUCE(8) - USAGE(3,9)/100 * PRODUCE(9) < 0;
! Stock limit restriction;
@SUM(PRODUCT(I)| I #EQ# 2 #OR# I #EQ# 5 #OR# I #EQ# 8: PRODUCE(I)) <= 600;
@SUM(PRODUCT(I)| I #EQ# 3 #OR# I #EQ# 6 #OR# I #EQ# 9: PRODUCE(I)) <= 800;
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  STOCK (kg):", @NEWLINE( 1));
@TABLE(STOCK);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE( J) #GT# 0: '  ',PRODUCT(J),' ',
        @FORMAT(PRODUCE(J),'%4.0f'),
                @IF(J #EQ# 5 #OR# J #EQ# 6 #OR# J #EQ# 8 #OR# J #EQ# 9, '-',' '),' kg  x Selling price p/kg: $',
        @FORMAT( PROFIT(J)/100  ,'%4.3f'),' = Total: $',
        @FORMAT( PROFIT(J)/100 * PRODUCE(J), '%3.0f'),
                @IF(J #EQ# 5 #OR# J #EQ# 6 #OR# J #EQ# 8 #OR# J #EQ# 9, '-',' '),
@NEWLINE(1));
@WRITE( '  Total:   ',
        @FORMAT(PRODUCE(5) + PRODUCE(6) + PRODUCE(8) + PRODUCE(9),'%4.0f'),'  kg (Stock used)',
@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
           ALLOY1_Y  ALLOY1_A  ALLOY1_B  ALLOY2_Y  ALLOY2_A  ALLOY2_B  ALLOY3_3  ALLOY3_A  ALLOY3_B
 SILICON   13.00000  15.00000  10.00000  14.00000  15.00000  10.00000  13.00000  15.00000  10.00000
 IRON      10.00000  13.00000  5.000000  11.00000  13.00000  5.000000  8.000000  13.00000  5.000000
 ALUMINUM  80.00000  72.00000  85.00000  78.00000  72.00000  85.00000  83.00000  72.00000  85.00000

PROFIT:
 ALLOY1_Y  8.000000
 ALLOY1_A  3.000000
 ALLOY1_B  5.000000
 ALLOY2_Y  7.000000
 ALLOY2_A  3.000000
 ALLOY2_B  5.000000
 ALLOY3_3  10.00000
 ALLOY3_A  3.000000
 ALLOY3_B  5.000000

STOCK (kg):
 ALLOY1_Y  0.000000
 ALLOY1_A  600.0000
 ALLOY1_B  800.0000
 ALLOY2_Y  0.000000
 ALLOY2_A  600.0000
 ALLOY2_B  800.0000
 ALLOY3_3  0.000000
 ALLOY3_A  600.0000
 ALLOY3_B  800.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                     52.00000
Infeasibilities:                                     0.000000

IDEAL PLANNING PROGRAM:
ALLOY2_Y 1000  kg  x Selling price p/kg: $0.070 = Total: $ 70
ALLOY2_A  538- kg  x Selling price p/kg: $0.030 = Total: $ 16-
ALLOY2_B  462- kg  x Selling price p/kg: $0.050 = Total: $ 23-
ALLOY3_3  400  kg  x Selling price p/kg: $0.100 = Total: $ 40
ALLOY3_A   62- kg  x Selling price p/kg: $0.030 = Total: $  2-
ALLOY3_B  338- kg  x Selling price p/kg: $0.050 = Total: $ 17-
Total:   1400  kg (Stock used)
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Alloy
- Mineral
- Chemical
- Blend

Source:

- Book 5
- Page 235

GOAL

The Pittsburgh Steel (PS) Co. has been contracted to produce a new type of very high carbon steel which has the following tight quality requirements: Carbon Content, Chrome Content , Manganese Content and Silicon Content, PS has the following materials available for mixing up a batch:

```
At Least (%):
 CARBON     0.0300000
 CHROME     0.0030000
 MANGANESE  0.0135000
 SILICON    0.0270000

Not More Than (%):
 CARBON     0.0350000
 CHROME     0.0045000
 MANGANESE  0.0165000
 SILICON    0.0300000
```

A one-ton (2000-Lb.) batch must be blended, which satisfies the quality requirements stated earlier. The problem now is what amounts of each of the eleven materials should be blended together to minimize the cost, but satisfy the quality requirements.

An experienced steel man claims the least cost mix will not use any more than nine of the eleven available raw materials. What is a good blend?

Most of the eleven prices and four quality control requirements are negotiable. Which prices and requirements are worth negotiating?

Note the chemical content of a blend is simply the weighted average of the chemical content of its components. Thus, for example, if we make a blend of 40% Alloy 1 and 60% Alloy 2, the manganese content is $(0.40) \times 60 + (0.60) \times 9 = 29.4$.

Formulation and Solution of the Pittsburgh Steel Blending Problem. The PS blending problem can be formulated as an LP with 11 variables and 13 constraints. The 11 variables correspond to the 11 raw materials from which we can choose.

Four constraints are from the upper usage limits on silicon carbide and steels. Four of the constraints are from the lower quality limits. Another four constraints are from the upper quality limits.

The thirteenth constraint is the requirement that the weight of all materials used must sum to 2000 pounds. Notice only 7 of the 11 raw materials were used. In actual practice, this type of LP was solved on a twice-monthly basis by Pittsburgh Steel.

The purchasing agent used the first solution, including the reduced cost and dual prices, as a guide in buying materials. The second solution later in the month was mainly for the metallurgist's benefit in making up a blend from the raw materials actually on hand.

Suppose we can pump oxygen into the furnace. This oxygen combines completely with carbon to produce the gas CO_2, which escapes. The oxygen will burn off carbon at the rate of 12 pounds of carbon burned off for each 32 pounds of oxygen. Oxygen costs two cents a pound. If you reformulated the problem to include this additional option, would it change the decisions?

The oxygen injection option to burn off carbon is clearly uninteresting because, in the current solution, it is the lower bound constraint rather than the upper bound on carbon that is binding. Thus, burning off-carbon by itself, even if it could be done at no expense, would increase the total cost of the solution.

```
MODEL:

SETS:
LCOL /
LIM_RM_CB,          ! Raw Material available - Carbon;
LIM_RM_SI1,         ! Raw Material available - Silicon 1;
LIM_RM_SI2,         ! Raw Material available - Silicon 2;
LIM_RM_SI3,         ! Raw Material available - Silicon 3;
LIM_CB_MIN,         ! Quality requirements, Minimum Carbon;
LIM_CB_MAX,         ! Quality requirements, Maximum Carbon;
LIM_CH_MIN,         ! Quality requirements, Minimum Chrome;
LIM_CH_MAX,         ! Quality requirements, Maximum Chrome;
LIM_MG_MIN,         ! Quality requirements, Minimum Manganese;
LIM_MG_MAX,         ! Quality requirements, Maximum Manganese;
LIM_SI_MIN,         ! Quality requirements, Minimum Silicon;
LIM_SI_MAX,         ! Quality requirements, Maximum Silicon;
LIM_FINISH          ! Finish good requirements;
/:LIMIT;

RESOURCE:
COST,               ! Cost per pound;
CARBON,             ! Percent Carbon;
CHROME,             ! Percent Chrome;
MANGANESE,          ! Percent Manganese;
SILICON,            ! Percent Silicon;
STOCK,              ! Amount Available;
PRODUCE;            ! Volume produced;

CONTENT:
AT_LEAST,           ! Quality requirements;
NOT_MORE_THAN;      ! Quality requirements;
ENDSETS

DATA:
! Resources attributes;
RESOURCE,  COST,    CARBON,  CHROME,  MANGANESE,  SILICON,  STOCK =
Pig_Iron1  0.0300   4.0      0.0      0.9         2.25      1.E30
Pig_Iron2  0.0645   0.0      10.0     4.5         15.00     1.E30
Ferro_SI1  0.0650   0.0      0.0      0.0         45.00     1.E30
Ferro_SI2  0.0610   0.0      0.0      0.0         42.00     1.E30
Alloy1     0.1000   0.0      0.0      60.0        18.00     1.E30
Alloy2     0.1300   0.0      20.0     9.0         30.00     1.E30
Alloy3     0.1190   0.0      8.0      33.0        25.00     1.E30
Carbide    0.0800   15.0     0.0      0.0         30.00     20
Steel1     0.0210   0.4      0.0      0.9         0.00      200
Steel2     0.0200   0.1      0.0      0.3         0.00      200
Steel3     0.0195   0.1      0.0      0.3         0.00      200;

! Requirements;
CONTENT,   AT_LEAST,  NOT_MORE_THAN =
Carbon     0.0300     0.0350
Chrome     0.0030     0.0045
Manganese  0.0135     0.0165
Silicon    0.0270     0.0300;
ENDDATA
```

```
SUBMODEL BLD:
[OBJ] MIN =  @SUM(RESOURCE(I): COST(I) * PRODUCE(I));
! Raw material;
LIMIT( 1) = STOCK( 8);
LIMIT( 2) = STOCK( 9);
LIMIT( 3) = STOCK(10);
LIMIT( 4) = STOCK(11);
! Content;
LIMIT( 5) = AT_LEAST(1) * 2000;
LIMIT( 6) = NOT_MORE_THAN(1) * 2000;
LIMIT( 7) = AT_LEAST(2) * 2000;
LIMIT( 8) = NOT_MORE_THAN(2) * 2000;
LIMIT( 9) = AT_LEAST(3)  * 2000;
LIMIT(10) = NOT_MORE_THAN(3) * 2000;
LIMIT(11) = AT_LEAST(4) * 2000;
LIMIT(12) = NOT_MORE_THAN(4) * 2000;
! Finish good requirements;
LIMIT(13) = 2000;
! Raw material availabilities;
[RMCB] PRODUCE(8)  <= STOCK(8);
[RMSI1] PRODUCE(9)  <= STOCK(9);
[RMSI2] PRODUCE(10) <= STOCK(10);
[RMSI3] PRODUCE(11) <= STOCK(11);
! Carbon content (Quality requirements on);
[CCBA]          @SUM(RESOURCE(I): CARBON/100*PRODUCE(I))    >= LIMIT( 5);
[CCBN]          @SUM(RESOURCE(I): CARBON/100*PRODUCE(I))    <= LIMIT( 6);
! Chrome content (Quality requirements on);
[CCHA]          @SUM(RESOURCE(I): CHROME/100*PRODUCE(I))    >= LIMIT( 7);
[CCHN]          @SUM(RESOURCE(I): CHROME/100*PRODUCE(I))    <= LIMIT( 8);
! Manganese content (Quality requirements on);
[CMGA]          @SUM(RESOURCE(I): MANGANESE/100*PRODUCE(I)) >= LIMIT( 9);
[CMGN]          @SUM(RESOURCE(I): MANGANESE/100*PRODUCE(I)) <= LIMIT(10);
! Silicon content (Quality requirements on);
[CSIA]          @SUM(RESOURCE(I): SILICON/100*PRODUCE(I))   >= LIMIT(11);
[CSIN]          @SUM(RESOURCE(I): SILICON/100*PRODUCE(I))   <= LIMIT(12);
! Finish good requirements (Quality requirements on);
[FGR] @SUM(RESOURCE(I):PRODUCE(I)) = LIMIT(13);
ENDSUBMODEL
CALC:
@SET('TERSEO',1);             ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);             ! Post status windows, 1 Yes, 0 No;
@SET('LINLEN',120);           ! Terminal page width (0:none);
@WRITE("  DATA:", @NEWLINE( 1), "  Cost per Pound:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  Percent Carbon:", @NEWLINE( 1));
@TABLE(CARBON);
@WRITE(" ", @NEWLINE( 1), "  Percent Chrome:", @NEWLINE( 1));
@TABLE(CHROME);
@WRITE(" ", @NEWLINE( 1), "  Percent Manganese:", @NEWLINE( 1));
@TABLE(MANGANESE);
@WRITE(" ", @NEWLINE( 1), "  Percent Silicon:", @NEWLINE( 1));
@TABLE(SILICON);
@WRITE(" ", @NEWLINE( 1), "  Amount Available (lb):", @NEWLINE( 1));
@TABLE(STOCK);
@WRITE(" ", @NEWLINE( 1), "  At Least (%):", @NEWLINE( 1));
@TABLE(AT_LEAST);
@WRITE(" ", @NEWLINE( 1), "  Not More Than (%):", @NEWLINE( 1));
@TABLE(NOT_MORE_THAN);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(BLD);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RESOURCE(I) | PRODUCE(I) #GT# 0: '  ',
        @FORMAT(RESOURCE(I),'-9s'), '  Used: ',
        @FORMAT(PRODUCE(I),'%6.1f'),' Pouds x Cost per Pound: $',
        @FORMAT(COST(I), '%7.5f'),' = Total: $',
        @FORMAT(COST(I) * PRODUCE(I),'%8.5f'),
@NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(BLD);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
Cost per Pound:
 PIG_IRON1  0.0300000
 PIG_IRON2  0.0645000
 FERRO_SI1  0.0650000
 FERRO_SI2  0.0610000
 ALLOY1     0.1000000
 ALLOY2     0.1300000
 ALLOY3     0.1190000
 CARBIDE    0.0800000
 STEEL1     0.0210000
 STEEL2     0.0200000
 STEEL3     0.0195000

Percent Carbon:
 PIG_IRON1   4.000000
 PIG_IRON2   0.000000
 FERRO_SI1   0.000000
 FERRO_SI2   0.000000
 ALLOY1      0.000000
 ALLOY2      0.000000
 ALLOY3      0.000000
 CARBIDE     15.00000
 STEEL1      0.400000
 STEEL2      0.100000
 STEEL3      0.100000

Percent Chrome:
 PIG_IRON1   0.000000
 PIG_IRON2   10.00000
 FERRO_SI1   0.000000
 FERRO_SI2   0.000000
 ALLOY1      0.000000
 ALLOY2      20.00000
 ALLOY3      8.000000
 CARBIDE     0.000000
 STEEL1      0.000000
 STEEL2      0.000000
 STEEL3      0.000000

Percent Manganese:
 PIG_IRON1   0.900000
 PIG_IRON2   4.500000
 FERRO_SI1   0.000000
 FERRO_SI2   0.000000
 ALLOY1      60.00000
 ALLOY2      9.000000
 ALLOY3      33.00000
 CARBIDE     0.000000
 STEEL1      0.900000
 STEEL2      0.300000
 STEEL3      0.300000

Percent Silicon:
 PIG_IRON1   2.250000
 PIG_IRON2   15.00000
 FERRO_SI1   45.00000
 FERRO_SI2   42.00000
 ALLOY1      18.00000
 ALLOY2      30.00000
 ALLOY3      25.00000
 CARBIDE     30.00000
 STEEL1      0.000000
 STEEL2      0.000000
 STEEL3      0.000000

Amount Available (lb):
 PIG_IRON1   0.10E+31
 PIG_IRON2   0.10E+31
 FERRO_SI1   0.10E+31
 FERRO_SI2   0.10E+31
 ALLOY1      0.10E+31
 ALLOY2      0.10E+31
 ALLOY3      0.10E+31
 CARBIDE     20.00000
 STEEL1      200.0000
 STEEL2      200.0000
 STEEL3      200.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                 59.55629
Infeasibilities:                                                 0.000000

IDEAL PLANNING PROGRAM:
PIG_IRON1  Used: 1474.3 Pouds x Cost per Pound: $0.03000 = Total: $44.22792
PIG_IRON2  Used:   60.0 Pouds x Cost per Pound: $0.06450 = Total: $ 3.87000
FERRO_SI2  Used:   22.1 Pouds x Cost per Pound: $0.06100 = Total: $ 1.34579
ALLOY1     Used:   14.2 Pouds x Cost per Pound: $0.10000 = Total: $ 1.42389
STEEL1     Used:  200.0 Pouds x Cost per Pound: $0.02100 = Total: $ 4.20000
STEEL2     Used:   29.4 Pouds x Cost per Pound: $0.02000 = Total: $ 0.58870
STEEL3     Used:  200.0 Pouds x Cost per Pound: $0.01950 = Total: $ 3.90000
```

7

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Formula
- Blend

Source:

- Book 5
- Page 236

GOAL

In this example, the Pittsburgh Steel Co. has been contracted to produce a new type of steel that has given tight quality requirements.

You have a given amount of materials available for mixing up a batch. A one-ton (2000 lb.) batch must be blended that satisfies the quality requirements stated earlier.

The problem now is what amounts of each of the eleven materials should be blended together so as to minimize the cost but satisfy the quality requirements.

A steel expert claims that the least cost mix will not use any more than nine of the eleven available raw materials. What is a good blend?

Most of the eleven cost and four quality control requirements are negotiable. Which cost and requirements are worth negotiating?

Note that the chemical content of a blend is simply the weighted average of the chemical content of its components.

Thus, for example: If we make a blend of 40 percent Alloy 1 and 60 percent Alloy 2, the manganese content is (0.40)*60+(0.60)*9=29.4.

The information used in the model is described below:

Formula		P1	P2	F1	F2	A1	A2	A3	CB	S1	S2	S3
Carbon	lb.	0.0400							0.1500	0.0040	0.0010	0.0010
Chrome	lb.		0.1000				0.2000	0.0800				
Manganese	lb.	0.0900	0.0450			0.8000	0.0900	0.3300		0.0090	0.0030	0.0030
Silicon	lb.	0.0225	0.1500	0.4500	0.4200	0.1800	0.3000	0.2500	0.3000			
Cost p/lb.	$	0.0300	0.0645	0.0650	0.6100	0.1000	0.1300	0.1190	0.0800	0.0210	0.0200	0.0195

Available		Min	Max
Carbon	lb.	60	70
Chrome	lb.	8	9
Manganese	lb.	27	33
Silicon	lb.	60	54

```
MODEL:
SETS:
  PRODUCT : COST, PRODUCE;
  RESOURCE: AVAILABLE_MIN, AVAILABLE_MAX;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Products attributes;
  PRODUCT,          COST   =
  P1                0.03
  P2                0.0645
  F1                0.065
  F2                0.061
  A1                0.1
  A2                0.13
  A3                0.119
  CB                0.08
  S1                0.021
  S2                0.02
  S3                0.0195;

! Resources attributes;
  RESOURCE,         AVAILABLE_MIN,     AVAILABLE_MAX   =
  CARBON            60                 70
  CHROME            8                  9
  MANGANESE         27                 33
  SILICON           54                 60;

! Required (%)      P1     P2     F1     F2     A1     A2     A3     CB     S1     S2     S3;
  USAGE =           0.04   0      0      0      0      0      0      0.15   0.004  0.001  0.001  ! CARBON;
                     0     0.1    0      0      0      0.2    0.08   0      0      0      0      ! CHROME;
                    0.009  0.045  0      0      0.8    0.09   0.33   0      0.009  0.003  0.003  ! MANGANESE;
                    0.0225 0.15   0.45   0.42   0.18   0.3    0.25   0.3    0      0      0 ;    ! SILICON;
ENDDATA
SUBMODEL MAX10:
  [OBJ]  MIN = @SUM( PRODUCT( p): COST( p) * PRODUCE( p));
! Finish good requirements;
@SUM( PRODUCT( p): PRODUCE( p)) = 2000;
! Raw material available;
  PRODUCE( 8)  <=  20;
  PRODUCE( 9)  <= 200;
  PRODUCE( 10) <= 200;
  PRODUCE( 11) <= 200;
! The minimum available constraints;
@FOR( RESOURCE( r):
        [AVA_MIN] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= AVAILABLE_MIN( r););
! The maximum available  constraints;
 @FOR( RESOURCE( r):
        [AVA_MAX] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= AVAILABLE_MAX( r););
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (lbs):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE_MIN (lbs):", @NEWLINE( 1));
@TABLE(AVAILABLE_MIN);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE_MAX (lbs):", @NEWLINE( 1));
@TABLE(AVAILABLE_MAX);
@WRITE(" ", @NEWLINE( 1), "  COST p/lbs:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX10);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL MIXING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT( J)| PRODUCE(J) #GT# 0: '  Raw: ',
        @FORMAT(PRODUCT( J),'-2s'),', Finish good:',
        @FORMAT(PRODUCE( J),'%8.3f'),' lbs x Unit cost: $',
        @FORMAT(COST( J),'%7.5f'), ' = Total: $',
        @FORMAT(COST( J) * PRODUCE( J),'%5.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
  FORMULA (lbs):
                        P1          P2          F1          F2
   CARBON          0.0400000   0.00000000   0.0000000   0.0000000
   CHROME          0.0000000   0.10000000   0.0000000   0.0000000
   MANGANESE       0.0090000   0.04500000   0.0000000   0.0000000
   SILICON         0.0225000   0.15000000   0.4500000   0.4200000

                        A1          A2          A3          CB
   CARBON          0.0000000   0.0000000   0.0000000   0.15000000
   CHROME          0.0000000   0.20000000  0.0800000   0.0000000
   MANGANESE       0.80000000  0.09000000  0.3300000   0.0000000
   SILICON         0.18000000  0.30000000  0.2500000   0.30000000

                        S1          S2          S3
   CARBON          0.00400000  0.00100000  0.0010000
   CHROME          0.0000000   0.0000000   0.0000000
   MANGANESE       0.00900000  0.00300000  0.0030000
   SILICON         0.0000000   0.0000000   0.0000000

  AVAILABLE_MIN (lbs):
   CARBON         60.000000
   CHROME         8.0000000
   MANGANESE      27.000000
   SILICON        54.000000

  AVAILABLE_MAX (lbs):
   CARBON         70.000000
   CHROME         9.0000000
   MANGANESE      33.000000
   SILICON        60.000000

  COST p/lbs:
   P1             0.0300000
   P2             0.0645000
   F1             0.0650000
   F2             0.6100000
   A1             0.1000000
   A2             0.1300000
   A3             0.1190000
   CB             0.0800000
   S1             0.0210000
   S2             0.0200000
   S3             0.0195000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                          59.86528
Infeasibilities:                                          0.000000


IDEAL MIXING PROGRAM:
Raw: P1, Finish good:1474.525 lbs x Unit cost: $0.03000 = Total: $44.24
Raw: P2, Finish good:  80.000 lbs x Unit cost: $0.06450 = Total: $ 5.16
Raw: F2, Finish good:  16.897 lbs x Unit cost: $0.06100 = Total: $ 1.03
Raw: A1, Finish good:   9.590 lbs x Unit cost: $0.10000 = Total: $ 0.96
Raw: S1, Finish good: 200.000 lbs x Unit cost: $0.02100 = Total: $ 4.20
Raw: S2, Finish good:  18.987 lbs x Unit cost: $0.02000 = Total: $ 0.38
Raw: S3, Finish good: 200.000 lbs x Unit cost: $0.01950 = Total: $ 3.90
```

BLOCK 13

Block: FERTILIZER

How to optimize the production of a certain type of fertilizer considering the availability of raw material, composition and incompatibilities, in order to obtain the lowest cost?

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Chemical
- Garner
- Blend

Source:

- Book 1
- Page 96

GOAL

A company wants to produce a certain type of fertilizer. For this, the rules to be considered in the mathematical model follow.

- The minimum operating capacity of each Garner is 30 kg;
- The raw materials are arranged in 8 Garner;
- Each mixer operates only with 3 Garner;
- Chemical incompatibility has been identified between certain raw materials, such as: G2 with G3, G3 with G7 and G7 with G8;
- The production should be 1000 kg of fertilizer.

The composition requires that the fertilizer have at least:

- 75 kg of Nitrogen;
- 30 kg of Phosphorus
- 15 kg of Potassium

Auxiliary variables BUILD(1) to BUILD(8) will be used to indicate whether or not the raw material enters the fertilizer composition;

To operate with 3 garner it is necessary: BUILD(1) + BUILD(2) + …. + BUILD(8) ≤ 3;

The minimum capacity of the silo should be expressed as: Garner 1: PRODUCE(1) - 30 * BUILD(1) > 0, and thus respectively for the other garner

The incompatibility between G(2) and BUILD(3) can be expressed thus: BUILD(2)+ BUILD(3) ≤ 1 and thus respectively for the other cases;

To indicate whether the raw material enters or not in the composition should be expressed as: PRODUCE(1)-1000 * BUILD(1) ≤ 0 and thus respectively for the other cases;

Process	Garner		G1	G2	G3	G3	G5	G6	G7	G8	Limit (kg)
	Operation		1	1	1	1	1	1	1	1	1000
Formula	Nitrogen	%	2	3		17		3	40	5	75
	Phosphor	%			2	5		1	1	3	30
	Potassium	%					6	1	4	3	15
Chemical Incompatibility				G2 and G3		G3 and G7		G7 and G8			-
Cost		$	12.00	16.00	8.00	26.00	16.00	8.00	11.00	19.00	-

```
MODEL:
SETS:
 RESOURCE: LIMIT;
 PRODUCT: COST, PRODUCE, BUILD;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE,        LIMIT          =
 NITROGEN         75
 PHOSPHOR         30
 POTASSIUM        15;
! Products attributes;
 PRODUCT ,        COST           =
 G1               12
 G2               16
 G3               8
 G4               26
 G5               16
 G6               8
 G7               11
 G8               19;
! Formula (%)     G1    G2    G3    G4    G5    G6    G7    G8;
 USAGE    =       2     3     0     17    0     3     40    5     ! NITROGEN;
                  0     0     2     5     0     1     1     3     ! PHOSPHOR;
                  0     0     0     0     6     1     4     3;    ! POTASSIUM;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN =   @SUM( PRODUCT(I): COST(I) * PRODUCE(I));
! Maximum production Limit;
[PRO]         @SUM(PRODUCT(I): PRODUCE(I)) = 1000;
! Limits;
[LIM1]        @SUM(PRODUCT(J):USAGE(1,J)/100 * PRODUCE(J)) >= LIMIT(1);
[LIM2]        @SUM(PRODUCT(J):USAGE(2,J)/100 * PRODUCE(J)) >= LIMIT(2);
[LIM3]        @SUM(PRODUCT(J):USAGE(3,J)/100 * PRODUCE(J)) >= LIMIT(3);
! Restriction on indicating whether or not raw material is in composition;
@FOR(PRODUCT(I): [MIX] PRODUCE(I) - 1000 * BUILD(I) <= 0);
! Operating restriction because the mixers are limited to 3 garner;
      [GAR3] @SUM(PRODUCT(I): BUILD(I)) <= 3;
! Minimum garner capacity ( 30kg);
@FOR(PRODUCT(I): [GCAP] PRODUCE(I) - 30 * BUILD(I) >= 0);
! Incompatibility between G2 and G3;
[INC1] BUILD(2) + BUILD(3) <= 1;
! Incompatibility between G3 and G7;
[INC2] BUILD(3) + BUILD(7) <= 1;
! Incompatibility between G7 and G8;
[INC3] BUILD(7) + BUILD(8) <= 1;
@FOR(PRODUCT(I):@BIN( BUILD));
ENDSUBMODEL;
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  COST P/KG:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  LIMIT (Kg):", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1),"  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  Garner: ',
      @FORMAT(PRODUCT(J),'-3s'), 'Produces:',
      @FORMAT(PRODUCE(J),'%4.0f'),'kg x Unit cost: $',
      @FORMAT(COST(J),'%5.2f'),' = Total: $' ,
      @FORMAT(COST(J) * PRODUCE(J),'%7.2f'),
@NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                  G1        G2        G3        G4        G5        G6        G7        G8
 NITROGEN   2.000000  3.000000  0.000000  17.00000  0.000000  3.000000  40.00000  5.000000
 PHOSPHOR   0.000000  0.000000  2.000000  5.000000  0.000000  1.000000  1.000000  3.000000
 POTASSIUM  0.000000  0.000000  0.000000  0.000000  6.000000  1.000000  4.000000  3.000000

COST P/KG:
 G1         12.00000
 G2         16.00000
 G3         8.000000
 G4         26.00000
 G5         16.00000
 G6         8.000000
 G7         11.00000
 G8         19.00000

LIMIT (Kg):
 NITROGEN   75.00000
 PHOSPHOR   30.00000
 POTASSIUM  15.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                        17800.00
Objective bound:                                        17800.00
Infeasibilities:                                        0.000000

IDEAL PLANNING PROGRAM:
Garner: G4 Produces: 300kg x Unit cost: $26.00 = Total: $7800.00
Garner: G6 Produces: 300kg x Unit cost: $ 8.00 = Total: $2400.00
Garner: G8 Produces: 400kg x Unit cost: $19.00 = Total: $7600.00
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Chemical
- Blend

Source:

- Book 2
- Page 229

GOAL

A company produces three types of fertilizers, from the mixture of nitrate, phosphate and potassium based ingredients and an inert component and the selling price of these type of fertilizers, as detailed below:

Resources / Products			Fertilizer Type			Components	
			FT1	FT2	FT3	Available (kg)	Cost
Proportional Component For Packing	Nitrate	%	5	5	10	1200	3,000.00
	Phosphate	%	10	10	10	2000	1,000.00
	Potassium	%	5	10	10	1400	1,800.00
	Inertial	%	80	75	70	1500	200.00
COST P/PACKAGE		$	34.40	40.00	52.00	-	
DEMAND		pac	650	470	355	-	
FIXED COST PER PACK (PROD+SALE)		$	300.00	300.00	300.00	-	
PRICE		4	800.00	960.00	1,100.00	-	

The cost of blending, packaging and sales promotion is estimated at $300 for any products. Determine the monthly output in order to maximize profit.

```
MODEL:
SETS:
  RESOURCE: AVAILABLE, PRODUCE;
  PRODUCT: PRICE, CPACK, CFIXE, DEMAND;
  RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
  RESOURCE ,        AVAILABLE   =
  NITRAT            1200
  PHOSPHATE         2000
  POTASSIUM         1400
  INERTIAL          1500;

! Products attributes;
  PRODUCT,          PRICE,      CPACK,      CFIXE,      DEMAND     =
  FT1               800         34          300         650
  FT2               960         40          300         470
  FT3               1100        52          300         355;

! Formula (%)       FT1         FT2         FT3;
  USAGE       =     5           5           10          ! NITRAT;
                    10          10          10          ! PHOSPHATE;
                    5           10          10          ! POTASSIUM;
                    80          75          70;         ! INERTIAL;
ENDDATA
SUBMODEL MAX2:
[OBJ] MAX =  @SUM( PRODUCT(I): PRICE(I) * PRODUCE(I)) -
             @SUM( PRODUCT(I): CFIXE(I) * PRODUCE(I)) -
             @SUM( PRODUCT(I): CPACK(I) * PRODUCE(I));
! The Demand Constraint;
@FOR(PRODUCT(I): [DEM] PRODUCE(I) = DEMAND(I));
! The Available Constraint;
@FOR( RESOURCE(I):[AVA] PRODUCE(I) <= AVAILABLE(I));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  SALES PRICE P/PAC:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  COST P/PAC:", @NEWLINE( 1));
@TABLE(CPACK);
@WRITE(" ", @NEWLINE( 1), "  FIXED COST P/PAC:", @NEWLINE( 1));
@TABLE(CFIXE);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE P/PAC:", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (PAC):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
        @FORMAT(PRODUCT(J),'-3s'), ' Price:$',
        @FORMAT(PRICE(J),'%4.0f'),' x ',
        @FORMAT(PRODUCE(J), '%3.0f'),'Pack = Revenue:$',
        @FORMAT(PRICE(J) * PRODUCE(J),'%6.0f'),' - Raw cost:$',
        @FORMAT(CPACK(J) * PRODUCE(J),'%5.0f'),' - Fixed cost:$',
        @FORMAT(CFIXE(J) * PRODUCE(J),'%5.0f'),' = Profit:$',
        @FORMAT(PRICE(J) * PRODUCE(J) - CPACK(J) * PRODUCE(J) - CFIXE(J) * PRODUCE(J), '%6.0f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                  FT1       FT2       FT3
 NITRAT     5.000000  5.000000  10.00000
 PHOSPHATE  10.00000  10.00000  10.00000
 POTASSIUM  5.000000  10.00000  10.00000
 INERTIAL   80.00000  75.00000  70.00000

SALES PRICE P/PAC:
 FT1        800.0000
 FT2        960.0000
 FT3        1100.000

COST P/PAC:
 FT1        34.00000
 FT2        40.00000
 FT3        52.00000

FIXED COST P/PAC:
 FT1        300.0000
 FT2        300.0000
 FT3        300.0000

AVAILABLE P/PAC:
 NITRAT     1200.000
 PHOSPHATE  2000.000
 POTASSIUM  1400.000
 INERTIAL   1500.000

DEMAND (PAC):
 FT1        650.0000
 FT2        470.0000
 FT3        355.0000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                                     859840.0
Infeasibilities:                                                                     0.000000

IDEAL PLANNING PROGRAM:
FT1 Price:$ 800 x 650Pack = Revenue:$520000 - Raw cost:$22100 - Fixed cost:$195000 = Profit:$302900
FT2 Price:$ 960 x 470Pack = Revenue:$451200 - Raw cost:$18800 - Fixed cost:$141000 = Profit:$291400
FT3 Price:$1100 x 355Pack = Revenue:$390500 - Raw cost:$18460 - Fixed cost:$106500 = Profit:$265540
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Chemical
- Blend

Source:

- Book 1
- Page 82

GOAL

A fertilizer manufacturer wants to produce 1000 kg of product at minimal cost.For this purpose it uses the raw materials of M1 ... M8, as well as the respective costs.

In the product obtained, the amounts of Nitrogen, Phosphorus and Potassium should be above the minimum requirements, quoted in MinReq.

The composition can be seen in Formula.

Formula		M1	M2	M3	M4	M5	M6	M7	M8	Min_Req (kg)
Nitrogen	%	2	3	0	17	0	2	4	5	75
Phosphor	%	0	0	2	5	0	1	1	3	30
Potassium	%	0	0	0	0	6	1	4	3	15
Cost (p/kg)	$	12.00	16.00	8.00	26.00	16.00	8.00	11.00	19.00	-

Develop the model that minimizes product costs.

```
MODEL:
SETS:
 RESOURCE: MINREQ;
 PRODUCT:  COST, DEMAND, PRODUCE;
 RXP( RESOURCE, PRODUCT): FORMULA;
ENDSETS
DATA:
! Minimum resources required attributes (kg);
 RESOURCE ,          MINREQ  =
 NITROGEN            75
 PHOSPHOR            30
 POTASSIUM           15;
! Products attributes (Raw Material);
 PRODUCT,            COST      =
 M1                  12
 M2                  16
 M3                  8
 M4                  26
 M5                  16
 M6                  8
 M7                  11
 M8                  19;
! Formula (%)  M1      M2      M3      M4      M5      M6      M7      M8;
 FORMULA    =        2       3       0       17      0       2       4       5    ! NITROGEN;
                     0       0       2       5       0       1       1       3    ! PHOSPHOR;
                     0       0       0       0       6       1       4       3;   ! POTASSIUM;
ENDDATA
SUBMODEL MIN3:
[OBJ] MIN = @SUM( PRODUCT(I): COST(I) * PRODUCE(I));
! The demand constraint;
@FOR(RESOURCE(I):
        @SUM( PRODUCT(J): PRODUCE(J)) = 1000;);
! Minimum required (kg);
@FOR(RESOURCE(I):
        @SUM(PRODUCT(J): FORMULA(I,J)/100 * PRODUCE(J)) >= MINREQ(I););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Resets line length;
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA (%):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  COST P/KG:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM REQUIRED (kg):", @NEWLINE( 1));
@TABLE(MINREQ);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN3);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1), "  Produce: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  . ',
        @FORMAT(PRODUCT(J),'-2s'), ' ',
        @FORMAT(PRODUCE(J), '%4.0f'),' kg   x  Unit cost: $',
        @FORMAT(COST(J),'%5.2f'),' =  Total: $',
        @FORMAT(COST(J) * PRODUCE(J),'%7.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA (%):
                 M1        M2        M3        M4        M5        M6        M7        M8
 NITROGEN   2.000000  3.000000  0.000000  17.00000  0.000000  2.000000  4.000000  5.000000
 PHOSPHOR   0.000000  0.000000  2.000000  5.000000  0.000000  1.000000  1.000000  3.000000
 POTASSIUM  0.000000  0.000000  0.000000  0.000000  6.000000  1.000000  4.000000  3.000000

COST P/KG:
 M1         12.00000
 M2         16.00000
 M3         8.000000
 M4         26.00000
 M5         16.00000
 M6         8.000000
 M7         11.00000
 M8         19.00000

MINIMUM REQUIRED (kg):
 NITROGEN   75.00000
 PHOSPHOR   30.00000
 POTASSIUM  15.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              16978.95
Infeasibilities:                              0.000000

IDEAL PLANNING PROGRAM:
Produce:
. M3   242 kg    x  Unit cost: $ 8.00 =  Total: $1936.84
. M4   326 kg    x  Unit cost: $26.00 =  Total: $8484.21
. M7   205 kg    x  Unit cost: $11.00 =  Total: $2257.89
. M8   226 kg    x  Unit cost: $19.00 =  Total: $4300.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Chemical
- Blend

GOAL

An industry produces three types of fertilizers, FTA, FTB and FTC, from the blend of ingredients based on Nitrate, Phosphate, Potassium and an inert component. The data for elaboration of the model follow the following:

Resources / Products		FTA	FTB	FTC	Available (ton)	Cost (p/ton)
Nitrate	%	10	10	20	2,200	4,000.00
Phosphor	%	8	8	8	3,000	2,000.00
Potassium	%	10	20	20	1,800	3,000.00
Inertial	%	72	62	52	infinite	300.00
Price	$	1,350.00	1,600.00	2,000.00	-	-

The company has a long-term contract for the monthly supply of 5000 tons of FTC fertilizer. Elaborate a linear programming model for production, maximizing profits.

```
MODEL:
SETS:
 RESOURCE: AVAILABLE, COST;
 PRODUCT:  PRICE, DEMAND, SCOST, PRODUCE;
 RXP( PRODUCT, RESOURCE): FORMULA;
ENDSETS
DATA:
! Minimum resources required attributes (kg);
 RESOURCE ,          AVAILABLE,    COST  =
 NITRATE             2200          4000
 PHOSPHOR            3000          2000
 POTASSIUM           1800          3000
 INERTIAL            1.E13         300;
! Products attributes ($/Tons);
 PRODUCT,            PRICE,        DEMAND      =
 FTA                 1350          0
 FTB                 1600          0
 FTC                 2000          5000;
! Formula (%)        NITRATE       PHOSPHOR           POTASSIUM          INERTIAL;
 FORMULA     =       10            8                  10                 72              ! FTA;
                     10            8                  20                 62              ! FTB;
                     20            8                  20                 52;             ! FTC;
ENDDATA
SUBMODEL MAX4:
[OBJ] MAX = @SUM( PRODUCT(I): PRICE(I) * PRODUCE(I)) - SCOST(1) - SCOST(2) - SCOST(3);
! Cost calculation;
@FOR(PRODUCT(I):
        SCOST(I) - @SUM(RESOURCE(K) : FORMULA(I,K)/100 * COST(K)) * PRODUCE(I) = 0);
! The Demand constraints;
@FOR(PRODUCT(I): PRODUCE(I) >= DEMAND(I));
! The Available Constraints  (Tons);
@FOR(RESOURCE(I)| I #LE# 3:
        @SUM(PRODUCT(K) : FORMULA(K,I)/100 * PRODUCE(K)) <= AVAILABLE;);
ENDSUBMODEL
CALC:
@SET('TERSEO',1);    ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);    ! Post status windows, 1 Yes, 0 No;
@WRITE("  DATA:", @NEWLINE( 1), "  FORMULA = Product vs Raw material (%):", @NEWLINE( 1));
@TABLE(FORMULA);
@WRITE(" ", @NEWLINE( 1), "  COST p/tons:", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (tons):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  PRICE (P/tons):", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX4);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1), "  Produce: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J): '  . ',
        @FORMAT(PRODUCT(J),'-2s'), ' ',
        @FORMAT(PRODUCE(J),'%4.0f'),' tons   Price: $',
        @FORMAT(PRICE(J) * PRODUCE(J),'%11.2f'), '  -  Cost: $',
        @FORMAT(@IF(J #EQ# 1, SCOST(1),  @IF(J #EQ# 3, SCOST(3), SCOST(2))),'%11.2f' ) ,' = Profit: $',
        @FORMAT(PRICE(J) * PRODUCE(J) - @IF(J #EQ# 1, SCOST(1), @IF(J #EQ# 3, SCOST(3), SCOST(2))),'%10.2f'),
@NEWLINE( 1));
@WRITE('    Totals' , 10*' ', 'Price: $',
        @FORMAT(@SUM( PRODUCT(I): PRICE(I) * PRODUCE(I)),'%11.2f' ) , '  -  Cost: $',
        @FORMAT(SCOST(1)+SCOST(2)+SCOST(3), '%11.2f'), 3*' ', 'Profit: $',
        @FORMAT(OBJ,'%10.2f'),
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
FORMULA = Product vs Raw material (%):
             NITRATE   PHOSPHOR  POTASSIUM   INERTIAL
 FTA        10.00000   8.000000   10.00000   72.00000
 FTB        10.00000   8.000000   20.00000   62.00000
 FTC        20.00000   8.000000   20.00000   52.00000

COST p/tons:
 NITRATE    4000.000
 PHOSPHOR   2000.000
 POTASSIUM  3000.000
 INERTIAL   300.0000

AVAILABLE (tons):
 NITRATE    2200.000
 PHOSPHOR   3000.000
 POTASSIUM  1800.000
 INERTIAL   0.1000000E+14

PRICE (P/tons):
 FTA        1350.000
 FTB        1600.000
 FTC        2000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                  3612000.
Infeasibilities:                                                  0.000000

IDEAL PLANNING PROGRAM:
Produce:
. FTA 8000 tons   Price: $10800000.00  -  Cost: $ 8608000.00 = Profit: $2192000.00
. FTB    0 tons   Price: $        0.00  -  Cost: $        0.00 = Profit: $      0.00
. FTC 5000 tons   Price: $10000000.00  -  Cost: $ 8580000.00 = Profit: $1420000.00
  Totals          Price: $20800000.00  -  Cost: $17188000.00 = Profit: $3612000.00
```

BLOCK 14

Block: INVESTMENTS

How to plan an investment in stocks for example, considering all alternatives involving risks, yields, limits, government rules, in order to get the highest return?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Capital
- Profitability
- Funds

Source:

- Book 1
- Page 53

GOAL

Consider, for example, the case of an investment fund that has the following stock options considering the rules below:

- Investments are classified by categories: A, B and C
- Government restrictions on investment funds state that:
- No single investment may exceed 15% of the total fund capital;
- The total investment by category can't exceed 40%.
- The expectation should be equalized at 100%

Resources / Products		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
Investments Category		A	A	B	C	A	B	A	C	B	C
Profit	$	10.00	15.00	5.00	20.00	12.00	15.00	10.00	5.00	5.00	10.00

Create the template to calculate the best investment solution as well as the fund's profitability.

```
MODEL:
SETS:
 PRODUCT: PROFIT, INDEX, PRODUCE;
ENDSETS
DATA:
! Products attributes;
 PRODUCT,   INDEX,        PROFIT     =
 E1         1             10
 E2         1             15
 E3         2             5
 E4         3             20
 E5         1             12
 E6         2             15
 E7         1             10
 E8         3             5
 E9         2             5
 E10        3             10;
ENDDATA
SUBMODEL MAX1:
! Maximize profitability;
[OBJ] MAX =  @SUM( PRODUCT(I): PROFIT(I)/100 * PRODUCE(I));
[PER]        @SUM(PRODUCT(I): PRODUCE(I)) = 100;
!Investment cap by category;
! Category A;
[CAT_A]      @SUM(PRODUCT(I) | INDEX(I) #EQ# 1 : PRODUCE(I)) <= 40;
! Category B;
[CAT_B]      @SUM(PRODUCT(I) | INDEX(I) #EQ# 2 : PRODUCE(I)) <= 40;
! Category C;
[CAT_C]      @SUM(PRODUCT(I) | INDEX(I) #EQ# 3 : PRODUCE(I)) <= 40;
!Invest in up to 15%;
@FOR(PRODUCT(I): [INV] PRODUCE(I) <= 15);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE("  COMPANIES,INVESTMENT CATEGORY (A=1, B=2, C=3):", @NEWLINE( 1));
@TABLE(INDEX);
@WRITE(" ", @NEWLINE( 1), "  COMPANIES,EXPECTED PROFITABILITY (%):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I): '  CIA:',
      @FORMAT(PRODUCT(I),'-3s'), ' Category:', @IF(INDEX(I) #EQ# 1,'A',@IF(INDEX(I) #EQ# 2,'B','C')),', Expected Profitability:',
      @FORMAT(PROFIT(I),'%5.2f'), '% x Maximum Investment:',
      @FORMAT(PRODUCE(I),'%5.2f'), '% = Profitability:',
      @FORMAT(PRODUCE(I) * PROFIT(I)/100,'%5.2f'),'%',
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COMPANIES,INVESTMENT CATEGORY (A=1, B=2, C=3):
 E1   1.000000
 E2   1.000000
 E3   2.000000
 E4   3.000000
 E5   1.000000
 E6   2.000000
 E7   1.000000
 E8   3.000000
 E9   2.000000
 E10  3.000000

COMPANIES, EXPECTED PROFITABILITY (%):
 E1   10.00000
 E2   15.00000
 E3   5.000000
 E4   20.00000
 E5   12.00000
 E6   15.00000
 E7   10.00000
 E8   5.000000
 E9   5.000000
 E10  10.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                                                12.55000
Infeasibilities:                                                                                0.000000

IDEAL PLANNING PROGRAM:
CIA:E1  Category:A, Expected Profitability:10.00% x Maximum Investment: 0.00% = Profitability: 0.00%
CIA:E2  Category:A, Expected Profitability:15.00% x Maximum Investment:15.00% = Profitability: 2.25%
CIA:E3  Category:B, Expected Profitability: 5.00% x Maximum Investment: 0.00% = Profitability: 0.00%
CIA:E4  Category:C, Expected Profitability:20.00% x Maximum Investment:15.00% = Profitability: 3.00%
CIA:E5  Category:A, Expected Profitability:12.00% x Maximum Investment:15.00% = Profitability: 1.80%
CIA:E6  Category:B, Expected Profitability:15.00% x Maximum Investment:15.00% = Profitability: 2.25%
CIA:E7  Category:A, Expected Profitability:10.00% x Maximum Investment:10.00% = Profitability: 1.00%
CIA:E8  Category:C, Expected Profitability: 5.00% x Maximum Investment:10.00% = Profitability: 0.50%
CIA:E9  Category:B, Expected Profitability: 5.00% x Maximum Investment: 5.00% = Profitability: 0.25%
CIA:E10 Category:C, Expected Profitability:10.00% x Maximum Investment:15.00% = Profitability: 1.50%
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Capital
- Profitability
- Funds

Source:

- Book 1
- Page 100

GOAL

A person has $8000 to invest in up to 3 investment options (A,B,C), which will provide the return after 5 years.

Each of the investments can be done in multiples of $1000, and he wants to invest a maximum of $4000 in a given option. The expected returns are shown below in data.

Investments / Products		1	2	3	4
A	$	2,000.00	5,000.00	7,500.00	10,000.00
B	$	1,500.00	3,500.00	8,000.00	11,000.00
C	$	1,100.00	2,500.00	8,500.00	10,500.00
Value	$	1,000.00	2,000.00	3,000.00	4,000.00

Knowing that the investor can invest in any options, what should be the investment policy in order to have the highest overall return?

```
MODEL:
SETS:
 PRODUCT : APPLICATION;
 RESOURCE: ;
 RXP( RESOURCE, PRODUCT): RETURN,PRODUCE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE    =
 OPTION_A
 OPTION_B
 OPTION_C;
! Product attributes;
 PRODUCT,          APPLICATION    =
 APP_1             1000
 APP_2             2000
 APP_3             3000
 APP_4             4000;
! Return attributes APP_1         APP_2         APP_3         APP_4;
 RETURN     =      2000          5000          7500          10000         ! OPTION_A;
                   1500          3500          8000          11000         ! OPTION_B;
                   1100          2500          8500          10500;        ! OPTION_C;
ENDDATA
SUBMODEL MAX2:
! Maximize profitability;
[OBJ] MAX = @SUM( RXP(I,J): RETURN(I,J) * PRODUCE(I,J));
!Investment limit;
[LIM] @SUM( RXP(I,J): APPLICATION(J) * PRODUCE(I,J) )<= 8000;
@FOR( RXP(I,J): [INT] @GIN(PRODUCE(I,J)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RETURN OF INVESTMENTS:", @NEWLINE( 1));
@TABLE(RETURN);
@WRITE(" ", @NEWLINE( 1), "  INVESTMENT VALUE:", @NEWLINE( 1));
@TABLE(APPLICATION);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX2);
!  Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP(I,J) | PRODUCE(I,J) #GT# 0: '  ',
        @FORMAT(RESOURCE(I),'-6s'), ' vs ',
        @FORMAT(PRODUCT(J),'-5s'),': $',
        @FORMAT(RETURN(I,J),'%6.2f'), ' x',
        @FORMAT(PRODUCE(I,J),'%2.0f'), ' = $',
        @FORMAT(PRODUCE(I,J) * RETURN(I,J),'%8.2f'), ' Return - Applied: $',
        @FORMAT(APPLICATION(J) * PRODUCE(I,J),'%6.2f'), ' = Profit: $',
        @FORMAT(PRODUCE(I,J) * RETURN(I,J) - APPLICATION(J) * PRODUCE(I,J),'%8.2f'),
@NEWLINE( 1));
@WRITE('  Maximum return:', 19*' ','$', @FORMAT(OBJ,'%8.2f'),@NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RETURN OF INVESTMENTS:
              APP_1     APP_2     APP_3     APP_4
 OPTION_A  2000.000  5000.000  7500.000  10000.00
 OPTION_B  1500.000  3500.000  8000.000  11000.00
 OPTION_C  1100.000  2500.000  8500.000  10500.00

INVESTMENT VALUE:
 APP_1     1000.000
 APP_2     2000.000
 APP_3     3000.000
 APP_4     4000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                                22000.00
Objective bound:                                                                22000.00
Infeasibilities:                                                                0.000000


IDEAL PLANNING PROGRAM:
OPTION_A vs APP_2: $5000.00 x 1 = $ 5000.00 Return - Applied: $2000.00 = Profit: $ 3000.00
OPTION_C vs APP_3: $8500.00 x 2 = $17000.00 Return - Applied: $6000.00 = Profit: $11000.00
Maximum return:                   $22000.00
```

3

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Planning
- Profitability
- Funds
- Risk

Source:
- Book 2
- Page 239

GOAL

A particular financial institution wants to know how to invest $100,000 in five stock funds in order to maximize return on investment, where the Share Funds are: A, B, C, D and E.

Resources / Products			Share					Limit
Stock Funds			A	B	C	D	E	%
Option	Short Term	Y/N	N	Y	N	N	Y	50
	Long Term	Y/N	Y	N	Y	Y	N	50
	Hard Risk	Y/N	Y	N	N	Y	Y	50
	Free Charge (1)	Y/N	Y	Y	N	Y	N	30
	Free Charge (2)	Y/N	Y	Y	N	Y	N	40
Profit		%	95	80	90	90	90	-

RULES

- Invest, 50% of the money in short-term stocks
- Invest up to 50% in high risk stocks.
- Invest at least 30% of the shares free of charge. (1)
- Income from shares that are free of fees must be at least 40% of total interest (2)
- The volume of applications must be equalized in 100%

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: LIMIT;
 RXP( RESOURCE, PRODUCT): INDEX;
ENDSETS
DATA:
! Resource attributes;
 RESOURCE ,              LIMIT  =
 SHORT_TERM              50
 LONG_TERM               50
 HARD_RISK               50
 FREE_CHARGE_APP         30
 FREE_CHARGE_INT         40;
! Products attributes;
 PRODUCT,                PROFIT       =
 SHARE_A                 95
 SHARE_B                 80
 SHARE_C                 90
 SHARE_D                 90
 SHARE_E                 90;
! Index attributes       SHARE_A    SHARE_B    SHARE_C    SHARE_D    SHARE_E;
 INDEX          =        0          1          0          0          1      ! SHORT_TERM;
                         1          0          1          1          0      ! LONG_TERM;
                         1          0          0          1          1      ! HARD_RISK;
                         1          1          0          1          0      ! FREE_CHARGE_APP;
                         1          1          0          1          0;     ! FREE_CHARGE_INT;
ENDDATA
SUBMODEL MAX3:
 [OBJ] MAX =  @SUM( PRODUCT(I): PROFIT(I) * PRODUCE(I));
!Equalization of investment;
[EQU]          @SUM( PRODUCT(I): PRODUCE(I)) = 100;
! Invest 50% Short term stocks;
[SHO]          @SUM( PRODUCT(I) | INDEX(1,I) #EQ# 1: PRODUCE(I)) <= LIMIT(1);
! Invest 50% long term stocks;
[LON]          @SUM( PRODUCT(I) | INDEX(2,I) #EQ# 1: PRODUCE(I)) <= LIMIT(2);
! Limit high-risk applications above 50%;
[HRK]          @SUM( PRODUCT(I) | INDEX(3,I) #EQ# 1: PRODUCE(I)) <= LIMIT(3);
! Invest 30% or more in shares free of charges;
[FTX]          @SUM( PRODUCT(I) | INDEX(4,I) #EQ# 1: PRODUCE(I)) >= LIMIT(4);
! Free charge + INTEREST SHOULD BE UP TO 40%;
[F40]          @SUM( PRODUCT(I) | INDEX(5,I) #EQ# 1: PRODUCE(I)) >= LIMIT(5);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  STOCK FUNDS (1=Yes, 0=No):", @NEWLINE( 1));
@TABLE(INDEX);
@WRITE(" ", @NEWLINE( 1), "  PROFIT (%):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  LIMIT (%):", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX3);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
      @FORMAT(PRODUCT(J),'-7s'), ', Return:',
      @FORMAT(PRODUCE(J),'%3.0f'),'% x Investment:$100000.00 = Applied:$',
      @FORMAT(PRODUCE(J)/100 * 100000, '%7.2f'),' x Rate:',
      @FORMAT(PROFIT(j)/1000,'%5.3f'),'%',' = Profit:$',
      @FORMAT((PRODUCE(J)/1000 * 100000) * PROFIT(J)/100,'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
STOCK FUNDS (1=Yes, 0=No):
                   SHARE_A   SHARE_B   SHARE_C   SHARE_D   SHARE_E
 SHORT_TERM       0.000000  1.000000  0.000000  0.000000  1.000000
 LONG_TERM        1.000000  0.000000  1.000000  1.000000  0.000000
 HARD_RISK        1.000000  0.000000  0.000000  1.000000  1.000000
 FREE_CHARGE_APP  1.000000  1.000000  0.000000  1.000000  0.000000
 FREE_CHARGE_INT  1.000000  1.000000  0.000000  1.000000  0.000000

PROFIT (%):
 SHARE_A          95.00000
 SHARE_B          80.00000
 SHARE_C          90.00000
 SHARE_D          90.00000
 SHARE_E          90.00000

LIMIT (%):
 SHORT_TERM       50.00000
 LONG_TERM        50.00000
 HARD_RISK        50.00000
 FREE_CHARGE_APP  30.00000
 FREE_CHARGE_INT  40.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                        8900.000
Infeasibilities:                                                        0.000000

IDEAL PLANNING PROGRAM:
SHARE_A, Return: 20% x Investment:$100000.00 = Applied:$20000.00 x Rate:0.095% = Profit:$1900.00
SHARE_B, Return: 20% x Investment:$100000.00 = Applied:$20000.00 x Rate:0.080% = Profit:$1600.00
SHARE_C, Return: 30% x Investment:$100000.00 = Applied:$30000.00 x Rate:0.090% = Profit:$2700.00
SHARE_E, Return: 30% x Investment:$100000.00 = Applied:$30000.00 x Rate:0.090% = Profit:$2700.00
```

4

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Capital
- Profitability
- Funds
- Risk

GOAL

A company has to plan its spending on Research & Development. The information for developing the model is as follows:

Year / Projects		1	2	3	4	Limit
1	$	70,000	80,000	90,000	50,000	200,000
2	$	15,000	20,000	20,000	30,000	70,000
3	$		25,000		40,000	70,000
4	$	20,000	15,000	30,000		70,000
5	$	20,000	10,000	20,000	20,000	70,000
Profit	$	105,990	128,900	136,140	117,380	-

She pre-selected 4 projects and should choose among which she should prioritize due to budget constraints.

```
MODEL:
SETS:
 PRODUCT : PROFIT, PRODUCE;
 RESOURCE: LIMIT;
 RXP( RESOURCE, PRODUCT): CAPITAL;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,         LIMIT   =
 YEAR1             200000
 YEAR2             70000
 YEAR3             70000
 YEAR4             70000
 YEAR5             70000;
! Products attributes;
 PRODUCT,          PROFIT    =
 PROJ1             105990
 PROJ2             128900
 PROJ3             136140
 PROJ4             117380;
! Capital Attributes  PROJ1     PROJ2     PROJ3     PROJ4;
 CAPITAL      =       70000     80000     90000     50000      ! YEAR1;
                      15000     20000     20000     30000      ! YEAR2;
                       0        25000     0         40000      ! YEAR3;
                      20000     15000     30000     0          ! YEAR4;
                      20000     10000     20000     20000;     ! YEAR5;
ENDDATA
SUBMODEL MAX4:
[OBJ] MAX = @SUM( PRODUCT(I): PROFIT(I) * PRODUCE(I));
@FOR( RESOURCE(L):
        [LIM] @SUM(PRODUCT(C): CAPITAL(L,C) * PRODUCE(C)) <= LIMIT(L););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  CAPITAL REQUIRED:", @NEWLINE( 1));
@TABLE(CAPITAL);
@WRITE(" ", @NEWLINE( 1), "  PROFIT:", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  LIMIT:", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MAX4);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
       @FORMAT(PRODUCT(J),'-5s'), '  Yield rate: ',
       @FORMAT(PRODUCE(J),'%7.5f'),'% x Project value: $',
       @FORMAT(PROFIT(J), '%7.2f'),' = Profit: $',
       @FORMAT(PROFIT(j)*PRODUCE(J),'%9.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX4);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
CAPITAL REQUIRED:
             PROJ1     PROJ2     PROJ3     PROJ4
 YEAR1  70000.00  80000.00  90000.00  50000.00
 YEAR2  15000.00  20000.00  20000.00  30000.00
 YEAR3  0.000000  25000.00  0.000000  40000.00
 YEAR4  20000.00  15000.00  30000.00  0.000000
 YEAR5  20000.00  10000.00  20000.00  20000.00

PROFIT:
 PROJ1  105990.0
 PROJ2  128900.0
 PROJ3  136140.0
 PROJ4  117380.0

LIMIT:
 YEAR1  200000.0
 YEAR2  70000.00
 YEAR3  70000.00
 YEAR4  70000.00
 YEAR5  70000.00
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                   362625.0
Infeasibilities:                                                   0.000000

IDEAL PLANNING PROGRAM:
PROJ1  Yield rate: 1.10680% x Project value: $105990.00 = Profit: $117309.32
PROJ2  Yield rate: 0.71845% x Project value: $128900.00 = Profit: $ 92607.77
PROJ4  Yield rate: 1.30097% x Project value: $117380.00 = Profit: $152707.96
```

5

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Profitability
- Funds
- Risk

Source:

- Book 2
- Page 169

GOAL

A retired pension resource planning expert has just consulted with a client who estimated to have 750000 available to apply next month when he retires. They agreed to consider feasible the application of the resources in the actions of the following companies:

Products / Resources			Maximum Limit For Investing			
			25%	Maturity	50%	35%
Ranking	Company	Return	$	#Year	Y/N	Y/N
Excellent	ABC Chemical	8.65%	187,500	11	Y	N
Good	Alfa Industrial	9.50%	187,500	10	Y	Y
Reasonable	Aurora Foundry	10.0%	187,500	6	N	Y
Great	Optical Vision	8.75%	187,500	10	Y	N
Good	Delta System	9.25%	187,500	7	N	Y
Very Good	Union Bank	9.0%	187,500	13	Y	N

At where:

- Return: represents the annual interest estimate for each of the shares
- Maturity: indicates the deadline when the shares are to be redeemed
- Ranking: indicates the quality or risk of each action

Rules:

- Do not apply more than 25% in the same company
- At least 50% of the capital in long-term shares with a maturity of 10 years or more
- Although Alpha, Aurora and Delta have the highest return, no more than 35% of the capital can be invested in them

```
MODEL:
SETS:
 RESOURCE: PROFIT, LIMIT, PRODUCE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE ,            PROFIT,     LIMIT   =
 ABC_CHEMICAL          8.65        187500
 ALFA_INDUSTRIAL       9.50        187500
 AURORA_FOUNDRY        10.00       187500
 OPTICAL_VISION        8.75        187500
 DELTA_SYSTEM          9.25        187500
 UNION_BANK            9.00        187500;
ENDDATA
SUBMODEL MAX5:
! Maximize profitability;
[OBJ] MAX = @SUM( RESOURCE(I): PROFIT(I)/100 * PRODUCE(I));
! Amount to be invested;
[AMO]  @SUM( RESOURCE(I): PRODUCE(I)) = 750000;
! Guarantee that no more than 25% will be invested in the same stock;
@FOR( RESOURCE(I): [G25] PRODUCE(I) <= LIMIT(I));
! 50% of the receivables must be applied in shares with more than 10 years of deadlines;
PRODUCE(1) + PRODUCE(2) + PRODUCE(4) + PRODUCE(6) >= 375000;
! 35% of the resource available classified as good;
PRODUCE(2) + PRODUCE(3) + PRODUCE(5) <= 262500;
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  25% LIMIT OF $750000 PER CIA:", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  RETURN RATE BY CIA (%):", @NEWLINE( 1));
@TABLE(PROFIT);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX5);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RESOURCE(J) | PRODUCE(J) #GT# 0: '  ',
       @FORMAT(RESOURCE(J),'-17s'), 'Purchase: $',
       @FORMAT(PRODUCE(J),'%9.2f'),' x Return rate:',
       @FORMAT(PROFIT(J), '%5.2f'),'% = Profit: $',
       @FORMAT(PROFIT(J)/100 * PRODUCE(J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX5);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
25% LIMIT OF $750000 PER CIA:
 ABC_CHEMICAL     187500.0
 ALFA_INDUSTRIAL  187500.0
 AURORA_FOUNDRY   187500.0
 OPTICAL_VISION   187500.0
 DELTA_SYSTEM     187500.0
 UNION_BANK       187500.0

RETURN RATE BY CIA (%):
 ABC_CHEMICAL     8.650000
 ALFA_INDUSTRIAL  9.500000
 AURORA_FOUNDRY   10.00000
 OPTICAL_VISION   8.750000
 DELTA_SYSTEM     9.250000
 UNION_BANK       9.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                      68887.50
Infeasibilities:                                                      0.000000

IDEAL PLANNING PROGRAM:
ABC_CHEMICAL     Purchase: $112500.00 x Return rate: 8.65% = Profit: $ 9731.25
ALFA_INDUSTRIAL  Purchase: $ 75000.00 x Return rate: 9.50% = Profit: $ 7125.00
AURORA_FOUNDRY   Purchase: $187500.00 x Return rate:10.00% = Profit: $18750.00
OPTICAL_VISION   Purchase: $187500.00 x Return rate: 8.75% = Profit: $16406.25
UNION_BANK       Purchase: $187500.00 x Return rate: 9.00% = Profit: $16875.00
```

6

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Planning
- Rental
- Profitability
- Project

Source:

- Book 2
- Page 242

GOAL

An entrepreneur plans to build apartments for students in a location near a large Federal University.

Resources / Products			Bedroom Apartments			
			1	2	3	Limit
Apt	Size For Unit	m2	65	75	95	3,720
	Qty	un	15	22	10	40
	Total Area	m2	975	1,650	950	3,575
Rental		$	450	550	750	-

Four types of apartments are defined in the project: 1, 2 or 3 bedroom apartments. The apartments measure respectively: 65, 75 and 95 m2.

The entrepreneur believes that the building can’t have more than 15 units of 1 dormitory, of 22 of 2 and 10 of 3.

The law of zoning in the place does not allow the construction of buildings with more than 40 apartments, and the maximum area of construction is 3,720 m2.

The entrepreneur has made an agreement with a local broker for which he will rent 5 one bedroom units, 8 of two.

A Marketing study indicates 1 bedroom units for $450, 2 for $550 and 3 for $750 per month.

How many units of each type does the entrepreneur need to build so that the potential rent input is maximized?

```
MODEL:
SETS:
 RESOURCE: LIMIT;
 PRODUCT: RENTAL, EXPECTATION, PRODUCE;
 RXP( RESOURCE, PRODUCT): USAGE;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,        LIMIT   =
 GROUND_AREA      3720
 QUANT_APT        40
 USEFUL_AREA      3575;
! Product attributes;
 PRODUCT ,        RENTAL       EXPECTATION  =
 APT1BED          450          15
 APT2BED          550          22
 APT3BED          750          10;
! Attributes      APT1BED      APT2BED      APT3BED;
 USAGE       =    65           75           95           ! GROUND_AREA;
                  15           22           10           ! QUANT_APT;
                  975          1650         950;         ! USEFUL_AREA;
ENDDATA
SUBMODEL MAX6:
[OBJ] MAX = @SUM( PRODUCT(I): RENTAL(I) * PRODUCE(I));
! The building can not have more apartments than:;
@FOR( PRODUCT(I): [EXPEC] PRODUCE(I) <= EXPECTATION(I));
      !Restriction of useful area to be built;
      [USU] @SUM( PRODUCT(I): USAGE(1,I) * PRODUCE(I)) <= USEFUL_AREA;
      ! Maximum number of apartments to be built (Law);
      [LIM] @SUM( PRODUCT(I):  PRODUCE(I)) <= LIMIT(2);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  ATTRIBUTES (m2, Un, m2):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  RENTAL VALUE:", @NEWLINE( 1));
@TABLE(RENTAL);
@WRITE(" ", @NEWLINE( 1), "  EXPECTATION (apt):", @NEWLINE( 1));
@TABLE(EXPECTATION);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX6);
! Solution Report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J) | PRODUCE(J) #GT# 0: '  ',
      @FORMAT(PRODUCT(J),'-9s'), 'Rental value: $',
      @FORMAT(RENTAL(J),'%5.2f'),' x ',
      @FORMAT(PRODUCE(J), '%2.0f'),' Apartments = Revenue: $',
      @FORMAT(RENTAL(J)* PRODUCE(J),'%8.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX6);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
ATTRIBUTES (m2, Un, m2):
               APT1BED   APT2BED   APT3BED
 GROUND_AREA  65.00000  75.00000  95.00000
 QUANT_APT    15.00000  22.00000  10.00000
 USEFUL_AREA  975.0000  1650.000  950.0000

RENTAL VALUE:
 APT1BED      450.0000
 APT2BED      550.0000
 APT3BED      750.0000

EXPECTATION (apt):
 APT1BED      15.00000
 APT2BED      22.00000
 APT3BED      10.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                23200.00
Infeasibilities:                                0.000000

IDEAL PLANNING PROGRAM:
APT1BED  Rental value: $450.00 x  8 Apartments = Revenue: $ 3600.00
APT2BED  Rental value: $550.00 x 22 Apartments = Revenue: $12100.00
APT3BED  Rental value: $750.00 x 10 Apartments = Revenue: $ 7500.00
```

BLOCK 15

Block: CLINIC

How to optimize the demand of a Clinic that does blood tests and has available some equipment with different production capacity, and different costs?

OTHER AVAILABLE BLOCKS

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Health
- Laboratory
- Medical

Source:

- Book 2
- Page 241

GOAL

A particular hospital faces a problem in their fluid analysis laboratory. The laboratory has 3 machines to analyze any type of fluid.

Recently, the demand for blood testing has grown so much that the lab director is having difficulty finalizing the analyses of the samples received before the arrival of new samples.

The laboratory works with 5 types of blood, and any machine can be used to analyze any type of fluid, however, the time required by the machine depends on the type of blood to be analyzed.

Resources / Products			T1	T2	T3	T4	T5	Limit
Machine Time	A	min	3	4	4	5	3	440
	B	min	5	3	5	4	5	440
	C	min	2	5	3	3	4	440
Demand		un	80	75	80	120	60	-

Each machine can be used for 8 hours (440 min) daily.

Blood collected in one day is stored and analyzed the next day.

Thus, at the beginning of each day the director must determine how to allocate each type of blood in each of the machines.

On a given morning, the laboratory has performed 80 blood tests of type 1, 75 of type 2, 80 of type 3, 120 of type 4 and 60 of type 5.

The director of the laboratory wants to know how much analysis of each type of blood that should be made in each machine in minimized way to use them.

```
MODEL:
SETS:
 HEADER1 / PROD, LIMIT, VALUE /:;
 PRODUCT : DEMAND;
 RESOURCE: LIMIT;
 RXP( RESOURCE, PRODUCT) : MTIME, PRODUCE;
 HXP( RESOURCE, HEADER1) : SLASUR;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,          LIMIT          =
 EQUIP_1            440
 EQUIP_2            440
 EQUIP_3            440;
! Products attributes;
 PRODUCT ,          DEMAND         =
 T1                 80
 T2                 75
 T3                 80
 T4                 120
 T5                 60;
! Machine time      T1     T2     T3     T4     T5;
 MTIME        =     3      4      4      5      3      !  EQUIP_1;
                    5      3      5      4      5      !  EQUIP_2;
                    2      5      3      3      4;     !  EQUIP_3;
ENDDATA
PROCEDURE PROC_HXP(X):
@FOR(HXP(I,J): SLASUR(I,J) = 0);
@FOR(HXP(I,J)| PRODUCE(I,X)  #GT# 0:
          SLASUR(I,2)  = DEMAND(X);
          SLASUR(I,1)  = PRODUCE(I,X);
          SLASUR(I,3)  = SLASUR(I,1) - SLASUR(I,2));
@TABLE(SLASUR);
ENDPROCEDURE
SUBMODEL MIN1:
[OBJ] MIN = @SUM( RXP( I,J): MTIME(I,J) * PRODUCE( I,J));
! Restriction of capacity of equipment;
 [L1]    @SUM(PRODUCT(J): PRODUCE(1,J)* MTIME(1,J)) = LIMIT(1);
 [L2]    @SUM(PRODUCT(J): PRODUCE(2,J)* MTIME(2,J)) = LIMIT(2);
 [L3]    @SUM(PRODUCT(J): PRODUCE(3,J)* MTIME(3,J)) = LIMIT(3);
! Demand constraints by type of tests;
 [T1]    @SUM(RESOURCE(L): PRODUCE(L,1)) <= DEMAND(1);
 [T2]    @SUM(RESOURCE(L): PRODUCE(L,2)) <= DEMAND(2);
 [T3]    @SUM(RESOURCE(L): PRODUCE(L,3)) <= DEMAND(3);
 [T4]    @SUM(RESOURCE(L): PRODUCE(L,4)) <= DEMAND(4);
 [T5]    @SUM(RESOURCE(L): PRODUCE(L,5)) <= DEMAND(5);
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
@WRITE("  DATA:", @NEWLINE( 1), "  MACHINE TIME (min):", @NEWLINE( 1));
@TABLE(MTIME);
@WRITE(" ", @NEWLINE( 1), "  LIMIT: (min)", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  Demand (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN1);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP(I,J)| PRODUCE(I,J) # GT# 0: '  ',
          @FORMAT(RESOURCE(I), '-6s'), ' produce:', ' ',
          @FORMAT(PRODUCE( I,J),'%3.0f'), ' Test ',
          @FORMAT(PRODUCT(J),'-3s'), '   x',
          @FORMAT(MTIME(I,J), '%3.0f'),' min   =   ',
          @FORMAT(PRODUCE( I,J) * MTIME(I,J),'%3.0f'), ' min',
@NEWLINE( 1));
! Slack/Surplus Product report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND, PROD = T1 (un): ", @NEWLINE( 1)); PROC_HXP(1);
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND, PROD = T2 (un): ", @NEWLINE( 1)); PROC_HXP(2);
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND, PROD = T3 (un): ", @NEWLINE( 1)); PROC_HXP(3);
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND, PROD = T4 (un): ", @NEWLINE( 1)); PROC_HXP(4);
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND, PROD = T5 (un): ", @NEWLINE( 1)); PROC_HXP(5);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
MACHINE TIME (min):
               T1        T2        T3        T4        T5
 EQUIP_1  3.000000  4.000000  4.000000  5.000000  3.000000
 EQUIP_2  5.000000  3.000000  5.000000  4.000000  5.000000
 EQUIP_3  2.000000  5.000000  3.000000  3.000000  4.000000

LIMIT: (min)
 EQUIP_1  440.0000
 EQUIP_2  440.0000
 EQUIP_3  440.0000

Demand (un):
 T1       80.00000
 T2       75.00000
 T3       80.00000
 T4       120.0000
 T5       60.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
SOLUTION:
Global optimal solution found.
Objective value:                              1320.000
Infeasibilities:                              0.000000
Total solver iterations:                             5

IDEAL PLANNING PROGRAM:
EQUIP_1 produce:  88 Test T4    x  5 min   =   440 min
EQUIP_2 produce:  80 Test T1    x  5 min   =   400 min
EQUIP_2 produce:   8 Test T3    x  5 min   =    40 min
EQUIP_3 produce:  75 Test T2    x  5 min   =   375 min
EQUIP_3 produce:  16 Test T5    x  4 min   =    65 min

SLACK/SURPLUS LIMIT = DEMAND, PROD = T1 (un):
              PROD      LIMIT      VALUE
 EQUIP_1   0.000000   0.000000   0.000000
 EQUIP_2   80.00000   80.00000   0.000000
 EQUIP_3   0.000000   0.000000   0.000000

SLACK/SURPLUS LIMIT = DEMAND, PROD = T2 (un):
               PROD       LIMIT       VALUE
 EQUIP_1   0.000000   0.000000   0.000000
 EQUIP_2   0.000000   0.000000   0.000000
 EQUIP_3   75.00000   75.00000   0.000000

SLACK/SURPLUS LIMIT = DEMAND, PROD = T3 (un):
               PROD       LIMIT       VALUE
 EQUIP_1   0.000000   0.000000   0.000000
 EQUIP_2   8.000000   80.00000  -72.00000
 EQUIP_3   0.000000   0.000000   0.000000

SLACK/SURPLUS LIMIT = DEMAND, PROD = T4 (un):
               PROD       LIMIT       VALUE
 EQUIP_1   88.00000   120.0000  -32.00000
 EQUIP_2   0.000000   0.000000   0.000000
 EQUIP_3   0.000000   0.000000   0.000000

SLACK/SURPLUS LIMIT = DEMAND, PROD = T5 (un):
               PROD       LIMIT       VALUE
 EQUIP_1   0.000000   0.000000   0.000000
 EQUIP_2   0.000000   0.000000   0.000000
 EQUIP_3   16.25000   60.00000  -43.75000
```

2

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Health
- Surgery
- Medical

GOAL

A small clinic specializes in orthopedic and aesthetic surgeries. Orthopedic surgeries average $10, while aesthetics are the most sought after and earn $30.

Each surgery patient requires four 30-minute appointments, totaling 2 hours of office care. The clinic has consultation rooms that allow 36 hours a week.

The aesthetic surgeries last in average 2 hours, while the orthopedic ones last 1 hour. The clinic's surgical center can be used 24 hours a week.

To have permanent mobilization of the teams of surgeons it is necessary to perform at least 2 aesthetic and 4 orthopedic surgeries per week.

Resources / Products		Aesthetics	Orthopedic	Limit (hr/week)
Doctor's Appointments	hr	2	2	36
Surgery	un	2	1	24
Minimum Surgery (p/week)	un	2	4	-
Price	$	30.00	10.00	-

Elaborate the model that maximizes the prescription of the clinic.

```
MODEL:
SETS:
  PRODUCT : PRICE, MIN_WEEK, PRODUCE;
  RESOURCE: LIMIT;
  RXP( RESOURCE, PRODUCT) : USAGE;
ENDSETS
DATA:
! Resource attributes;
  RESOURCE,        LIMIT          =
  DOC_APP_HR       36
  SURGERY_UN       24;
! Products attributes;
  PRODUCT ,        PRICE,              MIN_WEEK    =
  AESTHETICS       30                  2
  ORTHOPEDIC       10                  4;
! Required         AESTHETICS          ORTHOPEDIC;
  USAGE       =    2                   2              ! DOC_APP_HR;
                   2                   1;             ! SURGERY_UN;
ENDDATA
SUBMODEL MAX2:
[OBJ] MAX = @SUM( PRODUCT( p): PRICE( p) * PRODUCE( p));
! Minimum week;
[MWE1]  PRODUCE(1) >= MIN_WEEK(1);
[MWE2]  PRODUCE(2) >= MIN_WEEK(2);
! Limit constraints;
@FOR( RESOURCE( r):
        [LIM] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) <= LIMIT( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE:", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  LIMIT:", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  MINIMUM SURGERY P/WEEK:", @NEWLINE( 1));
@TABLE(MIN_WEEK);
@WRITE(" ", @NEWLINE( 1), "  SURGERY PRICE:", @NEWLINE( 1));
@TABLE(PRICE);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(I)| PRODUCE(I) #GT# 0: '  ', PRODUCT(I), ': ',
        @FORMAT(PRODUCE( I),'%2.0f'),' Surgery x Price:$',
        @FORMAT(PRICE(I),'%5.2f'), ' =  Total:$',
        @FORMAT(PRODUCE( I) * PRICE( I), '%5.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
REQUIRED (HR) :
             AESTHETICS  ORTHOPEDIC
 DOC_APP_HR    2.000000    2.000000
 SURGERY_HR    2.000000    1.000000

LIMIT (hr/week):
 DOC_APP_HR    36.00000
 SURGERY_HR    24.00000

MINIMUM SURGERY P/WEEK:
 AESTHETICS    2.000000
 ORTHOPEDIC    4.000000

SURGERY PRICE:
 AESTHETICS    30.00000
 ORTHOPEDIC    10.00000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              340.0000
Infeasibilities:                              0.000000

IDEAL PLANNING PROGRAM:
AESTHETICS: 10 Surgery x Price:$30.00 = Total: $300.00
ORTHOPEDIC:  4 Surgery x Price:$10.00 = Total: $ 40.00
```

3

GOAL

The Ministry of Health has five possible locations for the installation of Health Centers to serve four Population Centers.

An index has been constructed which expresses the inconvenience of serving each Population Center by the Health Center of each location, taking into account the number of inhabitants served and the means of transportation available.

The results are shown in the table of contents.

Population Center	Health Centers. (Inconvenience Rate)					Inhabitants
	HC1	HC2	HC3	HC4	HC5	
PC1	40	43	42	38	45	2,500
PC2	37	40	41	44	36	2,000
PC3	40	42	39	37	38	3,000
PC4	45	40	39	42	41	3,000

Suppose now that each Health Center can serve more than one Population Center, provided that the total number of inhabitants of Population Centers assigned to the same Health Center does not exceed 5,000. The number of inhabitants of Population Centers are listed in the table above.

Knowing that all population of the same Population Center has to be served by the same Health Center and that each Health Center serves only a Population Center, formulate a solution in LP.

Blocks
- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:
- Health
- Surgery
- Medical

Source:
- Book 12
- Page 13

```
MODEL:
SETS:
HEALT_CENTER:;
POPULATION_CENTER:INHABITANTS;
RXP(POPULATION_CENTER,HEALT_CENTER):INDEX, PRODUCE;
ENDSETS
DATA:
HEALT_CENTER =
HC1
HC2
HC3
HC4
HC5;
POPULATION_CENTER,      INHABITANTS  =
PC1                     2500
PC2                     2000
PC3                     3000
PC4                     3000;
! Inconvenience rate    HC1   HC2   HC3   HC4   HC5;
INDEX =                 40    43    42    38    45     !PC1;
                        37    40    41    44    36     !PC2;
                        40    42    39    37    38     !PC3;
                        45    40    39    42    41;    !PC4;
ENDDATA
SUBMODEL MIN3:
! Minimum Inconvenience Index;
[MII] MIN = @SUM(RXP(I,J): INDEX(I,J) * PRODUCE(I,J));
! A population center can only be served by a health center;
@FOR(POPULATION_CENTER(I):
        [PC] @SUM(HEALT_CENTER(J):PRODUCE(I,J)) = 1;);
!Each health center can only serve a population center;
@FOR(HEALT_CENTER(J):
        [HC] @SUM(POPULATION_CENTER(I): PRODUCE(I,J)) <= 1;);
!Each health center can serve more than one population center, provided that it does not have more than 5000 inhabitants.;
@FOR(HEALT_CENTER(J):
        [IN] @SUM(POPULATION_CENTER(I): INHABITANTS(I) * PRODUCE(I,J)) <= 5000;);
        @FOR(RXP(I,J): @BIN(PRODUCE(I,J)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  Inconvenience rate: ", @NEWLINE( 1));
@TABLE(INDEX);
@WRITE(" ", @NEWLINE( 1), "  INHABITANTS:", @NEWLINE( 1));
@TABLE(INHABITANTS);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN3);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR(RXP(I, J) | PRODUCE(I,J) #GT# 0: '  Population Center: ',
        @FORMAT(POPULATION_CENTER( I),'-5s'), ' Health Center: ',
        @FORMAT(HEALT_CENTER(J),'-5s'), 'Index: ',
        @FORMAT(PRODUCE(I,J) * INDEX(I,J),'%2.0f'), ' ',
@NEWLINE( 1));
@WRITE('  Minimum Inconvenience Index: ', 22*' ', MII, @NEWLINE(2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
Inconvenience rate:
           HC1       HC2       HC3       HC4       HC5
 PC1  40.00000  43.00000  42.00000  38.00000  45.00000
 PC2  37.00000  40.00000  41.00000  44.00000  36.00000
 PC3  40.00000  42.00000  39.00000  37.00000  38.00000
 PC4  45.00000  40.00000  39.00000  42.00000  41.00000

INHABITANTS:
 PC1  2500.000
 PC2  2000.000
 PC3  3000.000
 PC4  3000.000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              152.0000
Objective bound:                              152.0000
Infeasibilities:                              0.000000

IDEAL PLANNING PROGRAM:
Population Center: PC1   Healt Center: HC1   Index: 40
Population Center: PC2   Healt Center: HC5   Index: 36
Population Center: PC3   Healt Center: HC4   Index: 37
Population Center: PC4   Healt Center: HC3   Index: 39
Minimum Inconvenience Index:                       152
```

BLOCK 16

Block: CLASSICS

How to solve classic problems like the backpack, the traveling salesman, etc. using mathematical optimization as a tool?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Load
- Knapsack
- Medical

Source:

- Book 6
- Page 819

GOAL

The knapsack model is a classic problem that uses binary variables. In this problem, you have a group of items you want to pack into your knapsack.

Unfortunately, the capacity of the knapsack is limited such that it is impossible to include all items. Each item has a certain value, or utility, associated with including it in the knapsack.

The problem is to find the subset of items to include in the knapsack that maximizes the total value of the load without exceeding the capacity of the knapsack.

Of course, the knapsack euphemism shouldn't lead one to underestimate the importance of this class of problem.

The "knapsack" problem can be applied to many situations. Some examples are vehicle loading, capital budgeting and strategic planning.

Items	Rank	Weight		Limit
Repellent	2	kg	1	≤ 10
Sunblock	9	kg	1	
Isotonic	3	kg	2	
Condensate Milk	8	kg	1	
Chocolate	1	kg	1	
Dry Fruits	6	kg	1	
Sandwich	4	kg	1	
Water	10	kg	3	

```
MODEL:
SETS:
        ITEMS:
        INCLUDE,
        WEIGHT,
        RATING;
        MUST_EAT_ONE( ITEMS );
ENDSETS
DATA:
        ITEMS                   WEIGHT   RATING =
        REPELLENT               1        2
        SUNBLOCK                1        9
        ISOTONIC                2        3
        CONDENSED_MILK          1        8
        CHOCOLATE               1        1
        DRY_FRUITS              1        6
        SANDWICH                1        4
        AWATER                  3        10;
        KNAPSACK_CAPACITY = 10;
ENDDATA
SUBMODEL MAX1:
MAX = @SUM( ITEMS: RATING * INCLUDE);
[CAP] @SUM(ITEMS: WEIGHT * INCLUDE) <= KNAPSACK_CAPACITY;
@FOR( ITEMS: @BIN( INCLUDE));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',3);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  WEIGHT (KNAPSACK_CAPACITY = 10):", @NEWLINE( 1));
@TABLE(WEIGHT);
@WRITE(" ", @NEWLINE( 1), "  RATING:", @NEWLINE( 1));
@TABLE(RATING);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MAX1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( ITEMS(J): '  Item: ',
        @FORMAT(ITEMS(J),'-14s'), ' Rating:',
        @FORMAT(RATING( J),'%2.0f'),' x ','selected:',
        @FORMAT(INCLUDE( J),'%2.0f'), ' include:',
        @FORMAT(RATING(J) * INCLUDE( J),'%2.0f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MAX1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
WEIGHT (KNAPSACK_CAPACITY = 10):
 REPELLENT       1.00
 SUNBLOCK        1.00
 ISOTONIC        2.00
 CONDENSED_MILK  1.00
 CHOCOLATE       1.00
 DRY_FRUITS      1.00
 SANDWICH        1.00
 AWATER          3.00

RATING:
 REPELLENT       2.00
 SUNBLOCK        9.00
 ISOTONIC        3.00
 CONDENSED_MILK  8.00
 CHOCOLATE       1.00
 DRY_FRUITS      6.00
 SANDWICH        4.00
 AWATER          10.0
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    42.0
Objective bound:                                    42.0
Infeasibilities:                                    0.00


IDEAL PLANNING PROGRAM:
Item: REPELLENT      Rating: 2 x selected: 1 include: 2
Item: SUNBLOCK       Rating: 9 x selected: 1 include: 9
Item: ISOTONIC       Rating: 3 x selected: 1 include: 3
Item: CONDENSED_MILK Rating: 8 x selected: 1 include: 8
Item: CHOCOLATE      Rating: 1 x selected: 0 include: 0
Item: DRY_FRUITS     Rating: 6 x selected: 1 include: 6
Item: SANDWICH       Rating: 4 x selected: 1 include: 4
Item: AWATER         Rating:10 x selected: 1 include:10
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Travel
- Distance

Source:

- Book 4
- Dial a Ride

GOAL

The problem of the hawker or traveling salesman is a classic example in Linear Programming. It consists of minimizing the total distance to be traveled along 15 cities.

The logic is complex but the model works very well and can be reused only by changing the contents of the matrix of distances between cities contained in it.

For more information you will need to do a thorough analysis of the template code, available

	Distance Between Cities (Start=0, Finish=n)														
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	0	69	39	49	5	59	19	40	33	81	9		8	37	33
C2	69	0	47	39	54	9	49	59	95	47	98	87	69	36	67
C3	41	47	0	90	67	63	4	49	93	48		94	52	19	41
C4	49	39	90	0	21	60	56	80	12	69	54	52	33	99	2
C5		54	67	21	0	32	22	61	85	47	56	75	59	42	63
C6	59	9	63	60	32	0	58	46	43	3	33	65	69	80	37
C7	19	49		56	22	58	0	82	97	15	78	82	64	33	26
C8	37	59	49	80	61	46	82	0	8	86	1	90	37	0	42
C9	33	95	93	12	85	43	97	8	0	63	34	81	21	56	
C10	82	47	48	69	47	3	15	86	63	0	88	69	33	47	90
C11	9	98	7	54	56	33	78		34	88	0	77	63	87	57
C12	21	87	94	52	75	65	82	90	81	69	77	0	46	30	63
C13	8	69	52	33	59	69	64	37		33	63	46	0	59	95
C14	38	36	21	99	42	80	33	0	56	47	87	30	59	0	10
C15	33	67	41		63	40	26	42	0	90	57	63	95	40	0
Lower Distance Between 15 Cities															

```
MODEL:
SETS:
 CITY: U;              ! U( I) = sequence no. of city;
 LINK( CITY, CITY):
      DIST,            ! The distance matrix;
      X;               ! X( I, J) = 1 if link I, J is in tour;
 ENDSETS
DATA:
 ! Distance matrix, it need not be symmetric;
 CITY=
C1  C2  C3  C4  C5  C6  C7  C8  C9  C10  C11  C12  C13  C14  C15;

 DIST=
0   69  39  49  5   59  19  40  33  81  9   21  8   37  33 ! C1;
69  0   47  39  54  9   49  59  95  47  98  87  69  36  67 ! C2;
41  47  0   90  67  63  4   49  93  48  7   94  52  19  41 ! C3;
49  39  90  0   21  60  56  80  12  69  54  52  33  99  2  ! C4;
5   54  67  21  0   32  22  61  85  47  56  75  59  42  63 ! C5;
59  9   63  60  32  0   58  46  43  3   33  65  69  80  37 ! C6;
19  49  4   56  22  58  0   82  97  15  78  82  64  33  26 ! C7;
37  59  49  80  61  46  82  0   8   86  1   90  37  0   42 ! C8;
33  95  93  12  85  43  97  8   0   63  34  81  21  56  0  ! C9;
82  47  48  69  47  3   15  86  63  0   88  69  33  47  90 !C10;
9   98  7   54  56  33  78  1   34  88  0   77  63  87  57 !C11;
21  87  94  52  75  65  82  90  81  69  77  0   46  30  63 !C12;
8   69  52  33  59  69  64  37  21  33  63  46  0   59  95 !C13;
38  36  21  99  42  80  33  0   56  47  87  30  59  0   40 !C14;
33  67  41  2   63  40  26  42  0   90  57  63  95  40  0 ; !C15;
ENDDATA
SUBMODEL MIN2:
N = @SIZE( CITY);
! Minimize total distance traveled;
[OBJ] MIN = @SUM( LINK(i,j): DIST(i,j) * X(i,j));
! Stop k must be entered exactly once For k = 1,  we return to 1 only from an odd numbered (drop off) stop;
  @SUM( CITY(i) | @mod(i,2) #NE# 0 #and# i #NE# 1: x(i,1)) = 1;
  @SUM( CITY(i) | @mod(i,2) #EQ# 0 : x(i,1)) = 0;
  @FOR( CITY(k) | k #GT# 1:
        @SUM( CITY( i)| i #NE# k: X( i, k)) = 1;);
 ! Stop k must be departed exactly once For k = 1, we go first to an even numbered (pick up) stop;
  @SUM( CITY(j) | @mod(j,2) #EQ# 0 : x(1,j)) = 1;
  @SUM( CITY(j) | @mod(j,2) #NE# 0 : x(1,j)) = 0;
  @FOR( CITY(k) | k #GT# 1:
         @SUM( CITY( j)| j #NE# k: X( k, j)) = 1;
        X( k, k) = 0; );                         ! Cannot go from k to k;
 U(1) = 0;                                       ! Start empty;
! The case either i or j = 1;
 @FOR( CITY(i) | i #GT# 1:
        U(i) >= 2 - X(1,i) + (N-3)*X(i,1);
         U(i) <= (N-2) + X(i,1) - (N-3)*X(1,i););
 ! The case i,j > 1,
 ! This constraint, plus its "mirror", when i and j are switched,
  forces U(j) - U(i) = 1 if X(i,j) = 1;
  @FOR( LINK(i,j)| i #GT# 1 #AND# j #GT# 1 #AND# i #NE# j:U( j) >= U( i) + X(i,j) - X(j,i) - (N-2)*(1 - X(i,j) - X(j,i)););
 ! Make the X's 0/1;
  @FOR( LINK(i,j): @BIN( X(i,j)););
  ! The dial-a-ride feature.  Every pickup occurs before its drop-off;
  @FOR(CITY(i) | @mod(i,2)#EQ#0 :  U(i) <= U(i+1));
 ! Some cuts.  We know the sum of the stop numbers;
   @SUM( CITY( i): U( i)) = ( N-1)*N/2;
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',3);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  THE DISTANCE MATRIX (mile):", @NEWLINE( 1));
@TABLE(DIST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution report
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( LINK(I,J) | X(I,J) #GT# 0: '  From: ',
        @FORMAT(CITY(I),'-4s'),'        To: ',
        @FORMAT(CITY(J),'-4s'),'          Distance:',
        @FORMAT(DIST(i,j) * X(i,j),'%4.0f'),' mile',
@NEWLINE( 1));
@WRITE('  Total distance:',30*' ', @FORMAT(OBJ, '%4.0f'), ' mile', @ NEWLINE(1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

DATA:
THE DISTANCE MATRIX (mile):

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	0.00	69.0	39.0	49.0	5.00	59.0	19.0	40.0	33.0	81.0	9.00	21.0	8.00	37.0	33.0
C2	69.0	0.00	47.0	39.0	54.0	9.00	49.0	59.0	95.0	47.0	98.0	87.0	69.0	36.0	67.0
C3	41.0	47.0	0.00	90.0	67.0	63.0	4.00	49.0	93.0	48.0	7.00	94.0	52.0	19.0	41.0
C4	49.0	39.0	90.0	0.00	21.0	60.0	56.0	80.0	12.0	69.0	54.0	52.0	33.0	99.0	2.00
C5	5.00	54.0	67.0	21.0	0.00	32.0	22.0	61.0	85.0	47.0	56.0	75.0	59.0	42.0	63.0
C6	59.0	9.00	63.0	60.0	32.0	0.00	58.0	46.0	43.0	3.00	33.0	65.0	69.0	80.0	37.0
C7	19.0	49.0	4.00	56.0	22.0	58.0	0.00	82.0	97.0	15.0	78.0	82.0	64.0	33.0	26.0
C8	37.0	59.0	49.0	80.0	61.0	46.0	82.0	0.00	8.00	86.0	1.00	90.0	37.0	0.00	42.0
C9	33.0	95.0	93.0	12.0	85.0	43.0	97.0	8.00	0.00	63.0	34.0	81.0	21.0	56.0	0.00
C10	82.0	47.0	48.0	69.0	47.0	3.00	15.0	86.0	63.0	0.00	88.0	69.0	33.0	47.0	90.0
C11	9.00	98.0	7.00	54.0	56.0	33.0	78.0	1.00	34.0	88.0	0.00	77.0	63.0	87.0	57.0
C12	21.0	87.0	94.0	52.0	75.0	65.0	82.0	90.0	81.0	69.0	77.0	0.00	46.0	30.0	63.0
C13	8.00	69.0	52.0	33.0	59.0	69.0	64.0	37.0	21.0	33.0	63.0	46.0	0.00	59.0	95.0
C14	38.0	36.0	21.0	99.0	42.0	80.0	33.0	0.00	56.0	47.0	87.0	30.0	59.0	0.00	40.0
C15	33.0	67.0	41.0	2.00	63.0	40.0	26.0	42.0	0.00	90.0	57.0	63.0	95.0	40.0	0.00

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                  212.
Objective bound:                                  212.
Infeasibilities:                                  0.00

IDEAL PLANNING PROGRAM:
From: C1          To: C12          Distance:  21 mile
From: C2          To: C6           Distance:   9 mile
From: C3          To: C11          Distance:   7 mile
From: C4          To: C5           Distance:  21 mile
From: C5          To: C1           Distance:   5 mile
From: C6          To: C10          Distance:   3 mile
From: C7          To: C3           Distance:   4 mile
From: C8          To: C13          Distance:  37 mile
From: C9          To: C15          Distance:   0 mile
From: C10         To: C7           Distance:  15 mile
From: C11         To: C8           Distance:   1 mile
From: C12         To: C14          Distance:  30 mile
From: C13         To: C9           Distance:  21 mile
From: C14         To: C2           Distance:  36 mile
From: C15         To: C4           Distance:   2 mile
Total distance:                               212 mile
```


TRAVELING SALESPERSON PROBLEM
Lower distance between 15 cities / Sequence / Total distance: 212
21
C12
30
C14
36
C2
9
C6
3
C10
15
C13
37
C8
1
C11
7
C3
4
C7
21
C9
0
C15
2
C4
21
C5
5
C1

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Travel
- Distance

Source:

- Book 4
- TCPCUTX

GOAL

The problem of traveling salesman is a classic example in Linear Programming. In this case simple version.(TSPCUTx).

Sub-tour elimination method. We add a row/cut/constraint to cut off a sub-tour until a full tour is found. Find the shortest route that visits each city exactly once.

	Distance Between Cities (Start=0, Finish=n)											
FROM	NYK	ATL	CHI	CIN	HOU	LAX	MON	PHL	PIT	STL	SND	SNF
NYK	0	864	845	664	1706	2844	396	92	386	1002	2892	3032
ATL	864	0	702	454	842	2396	1196	772	714	554	2363	2679
CHI	845	702	0	324	1093	2136	764	764	459	294	2184	2187
CIN	664	454	324	0	1137	2180	798	572	284	338	2228	2463
HOU	1706	842	1093	1137	0	1616	1857	1614	1421	799	1521	2021
LAX	2844	2396	2136	2180	1616	0	2900	2752	2464	1842	95	405
MON	396	1196	764	798	1857	2900	0	424	514	1058	2948	2951
PHL	92	772	764	572	1614	5752	424	0	305	910	2800	2951
PIT	386	714	459	284	1421	2464	514	305	0	622	2512	2646
STL	1002	554	294	338	799	1842	1058	910	622	0	1890	2125
SND	2892	2363	2184	2228	1521	95	2948	2800	2800	2512	0	500
SNF	3032	2679	2187	2463	2021	405	2951	2951	2646	2125	500	0

```
MODEL:
SETS:
 CITY;
 LINK( CITY, CITY):
     DIST,              ! The distance matrix;
        Y;              ! Y( I, J) = 1 if link I, J is in tour;
 SUBTOUR: TOURSIZE;
 SXC(SUBTOUR,CITY): FLAG;
ENDSETS
DATA:
 ! Max sub-tour cuts allowed;
 SUBTOUR = 1..30;
CITY =
 NYK   ATL   CHI   CIN   HOU   LAX   MON   PHL   PIT   STL   SND   SNF;
  ! Distance matrix, it need not be symmetric;
 DIST =
 0     864   845   664   1706  2844  396   92    386   1002  2892  3032  !NYK;
 864   0     702   454   842   2396  1196  772   714   554   2363  2679  !ATL;
 845   702    0    324   1093  2136  764   764   459   294   2184  2187  !CHI;
 664   454   324   0     1137  2180  798   572   284   338   2228  2463  !CIN;
 1706  842   1093  1137   0    1616  1857  1614  1421  799   1521  2021  !HOU;
 2844  2396  2136  2180  1616   0    2900  2752  2464  1842  95    405   !LAX;
 396   1196  764   798   1857  2900  0     424   514   1058  2948  2951  !MON;
 92    772   764   572   1614  2752  424   0     305   910   2800  2951  !PHL;
 386   714   459   284   1421  2464  514   305   0     622   2512  2646  !PIT;
 1002  554   294   338   799   1842  1058  910   622   0     1890  2125  !STL;
 2892  2363  2184  2228  1521  95    2948  2800  2512  1890   0    500   !SND;
 3032  2679  2187  2463  2021  405   2951  2951  2646  2125  500    0;   !SNF;
ENDDATA
SUBMODEL TSP_CUT:
! Minimize total distance traveled;
 [OBJ] MIN = TOURLEN;
   TOURLEN = @SUM( LINK: DIST * Y);
! The Assignment constraints;
 @FOR( CITY( K):
          !  It must be entered;
         @SUM( CITY( I)| I #NE# K: Y( I, K)) = 1;
         !  It must be departed;
          @SUM( CITY( J)| J #NE# K: Y( K, J)) = 1;
         !  Cannot visit self;
         Y( K, K) = 0   );
! Sub-tour cuts added so far;
 @FOR( SUBTOUR(t) | t #LE# ICUT:
         @SUM( CITY(I) | FLAG(t,i) #EQ# 1:
         @SUM( CITY(J) | FLAG(t,j) #EQ# 1: Y(i,j))) <= TOURSIZE(t) - 1;);
 ! The Y(i,j) must be 0 or 1;
 @FOR( LINK(i,j): @BIN( Y(i,j)));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',5);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  THE DISTANCE MATRIX:", @NEWLINE( 1));
@TABLE(DIST);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 2));
 N = @SIZE( CITY);
! Max cuts we have space for;
 MXCUTS = @SIZE(SUBTOUR);
 ICUT = 0;
 MORECUTS = 1; ! =0 if no more cuts;
 @WHILE( MORECUTS :
! Loop over subtour cuts, ICUT;
!  Solve current version;
  @SOLVE( TSP_CUT);
  ICUT = ICUT + 1;
  @WRITE( @NEWLINE(1),'#',ICUT,' Obj value = ', TOURLEN, @NEWLINE(1));
  ! Turn this on to see intermediate solutions;
  !@FOR( LINK(i,j) | Y(i,j) #GT# .01: @WRITE(' ',i,' ',j,' ',Y(i,j),@NEWLINE(1)) );
!  Find subtour if any;
  StartCity = 1; ! Start at city 1;
  KURSTOP = StartCity;
  @FOR( CITY(k): FLAG(ICUT,k) = 0); ! Start with no cities in cut;
  @WRITE('  Begin tour  at     ', CITY(KURSTOP), @NEWLINE(1));
  TOURSIZE(ICUT) = 1;
!  Loop over cities KURSTOP to find subtour including 1;
  FLAG(ICUT, KURSTOP) = 1; ! City KURSTOP is in the cut;
  NotHome = 1;
  @WHILE( NotHome:
    @FOR(CITY(j) | Y(KURSTOP, j) #GT# 0.5:  ! Find next stop;
     @WRITE('  Next stop on tour= ', CITY(j),@NEWLINE(1));
         @IFC( j #EQ# startCity:
                        NotHome = 0;   ! Back home to startCity;
       @ELSE
                        TOURSIZE(ICUT) = TOURSIZE(ICUT) + 1;
                        KURSTOP = j;
                        FLAG(ICUT,KURSTOP) = 1; ! KURSTOP is in the cut;
          ); ! end of IFC;
      ); ! end of for loop;
   ); ! end of while loop;
   @IFC( TOURSIZE(ICUT) #EQ# N:
        @WRITE(' Above is complete tour, so min length tour found.');
          MORECUTS = 0;
       @ELSE
         @WRITE(' Add constraint to cut off above subtour.',@NEWLINE(1)); );
   @IFC( ICUT #EQ# @SIZE(SUBTOUR):
        @WRITE(' Exhausted memory.');
        MORECUTS = 0; );
   ); ! End loop over add cuts;
!  (There are various refinements/complications of this basic idea to make order of magnitude performance improvements);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(TSP_CUT);
ENDCALC
END
```

DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
THE DISTANCE MATRIX:
```

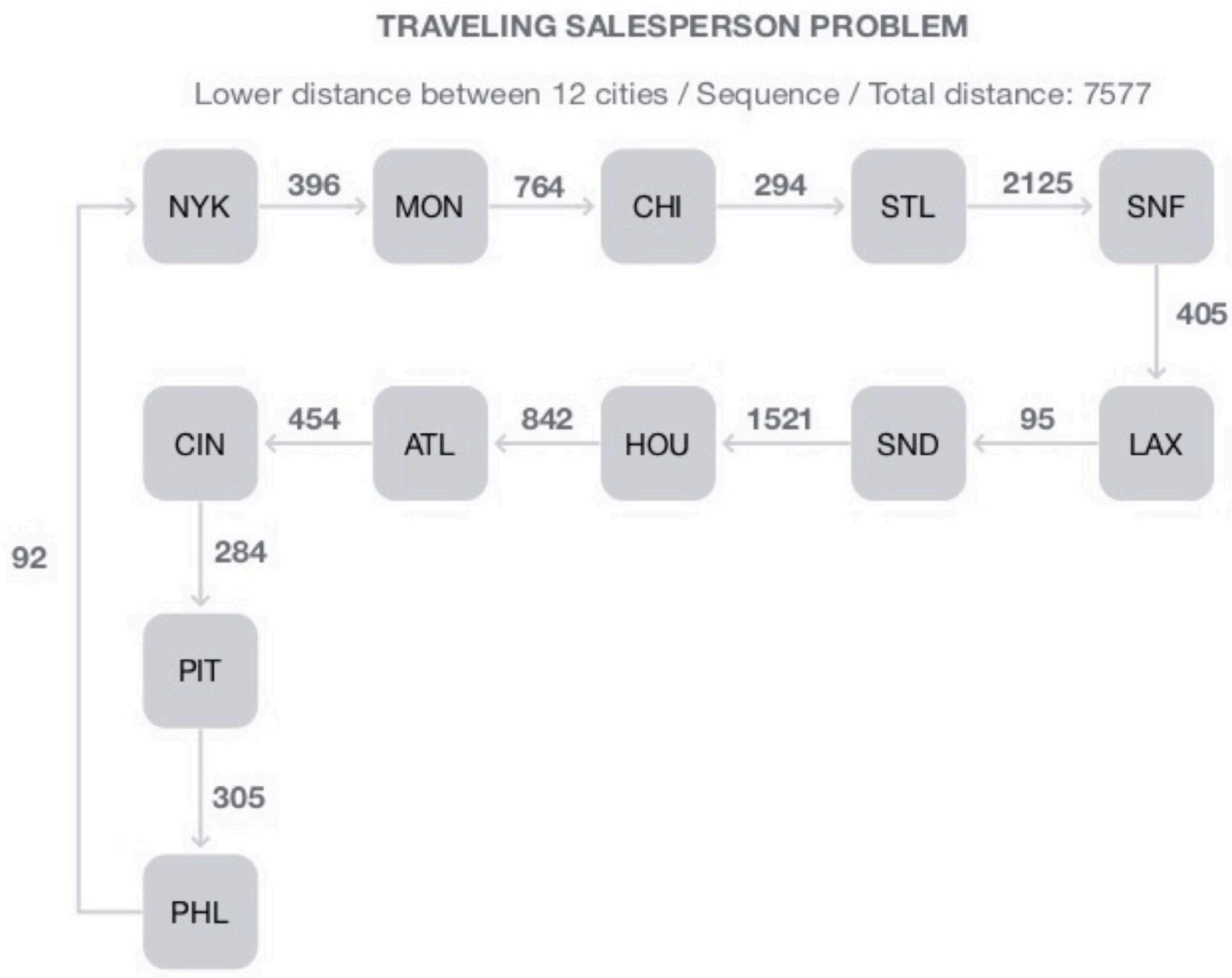
	NYK	ATL	CHI	CIN	HOU	LAX	MON	PHL	PIT	STL	SND	SNF
NYK	0.0000	864.00	845.00	664.00	1706.0	2844.0	396.00	92.000	386.00	1002.0	2892.0	3032.0
ATL	864.00	0.0000	702.00	454.00	842.00	2396.0	1196.0	772.00	714.00	554.00	2363.0	2679.0
CHI	845.00	702.00	0.0000	324.00	1093.0	2136.0	764.00	764.00	459.00	294.00	2184.0	2187.0
CIN	664.00	454.00	324.00	0.0000	1137.0	2180.0	798.00	572.00	284.00	338.00	2228.0	2463.0
HOU	1706.0	842.00	1093.0	1137.0	0.0000	1616.0	1857.0	1614.0	1421.0	799.00	1521.0	2021.0
LAX	2844.0	2396.0	2136.0	2180.0	1616.0	0.0000	2900.0	2752.0	2464.0	1842.0	95.000	405.00
MON	396.00	1196.0	764.00	798.00	1857.0	2900.0	0.0000	424.00	514.00	1058.0	2948.0	2951.0
PHL	92.000	772.00	764.00	572.00	1614.0	2752.0	424.00	0.0000	305.00	910.00	2800.0	2951.0
PIT	386.00	714.00	459.00	284.00	1421.0	2464.0	514.00	305.00	0.0000	622.00	2512.0	2646.0
STL	1002.0	554.00	294.00	338.00	799.00	1842.0	1058.0	910.00	622.00	0.0000	1890.0	2125.0
SND	2892.0	2363.0	2184.0	2228.0	1521.0	95.000	2948.0	2800.0	2512.0	1890.0	0.0000	500.00
SNF	3032.0	2679.0	2187.0	2463.0	2021.0	405.00	2951.0	2951.0	2646.0	2125.0	500.00	0.0000

SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Model Class: MILP
IDEAL PLANNING PROGRAM:
Global optimal solution found.
Objective value:                              4752.0
Objective bound:                              4752.0
Infeasibilities:                              0.0000

#1 Obj value = 4752
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              4852.0
  Objective bound:                              4852.0
  Infeasibilities:                              0.0000

 #2 Obj value = 4852
  Begin tour  at     NYK
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              4947.0
  Objective bound:                              4947.0
  Infeasibilities:                              0.0000

 #3 Obj value = 4947
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= PIT
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              5130.0
  Objective bound:                              5130.0
  Infeasibilities:                              0.0000
```

```
#5 Obj value = 5157
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= CHI
  Next stop on tour= STL
  Next stop on tour= CIN
  Next stop on tour= PIT
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              5230.0
  Objective bound:                              5230.0
  Infeasibilities:                              0.0000

#6 Obj value = 5230
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= CHI
  Next stop on tour= STL
  Next stop on tour= HOU
  Next stop on tour= ATL
  Next stop on tour= CIN
  Next stop on tour= PIT
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              6952.0
  Objective bound:                              6952.0
  Infeasibilities:                              0.0000

#7 Obj value = 6952
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= PIT
  Next stop on tour= CIN
  Next stop on tour= ATL
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              6995.0
  Objective bound:                              6995.0
  Infeasibilities:                              0.0000

#8 Obj value = 6995
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= CHI
  Next stop on tour= STL
  Next stop on tour= ATL
  Next stop on tour= CIN
  Next stop on tour= PIT
  Next stop on tour= PHL
  Next stop on tour= NYK
 Add constraint to cut off above subtour.
  Global optimal solution found.
  Objective value:                              7577.0
  Objective bound:                              7577.0
  Infeasibilities:                              0.0000

#9 Obj value = 7577
  Begin tour  at     NYK
  Next stop on tour= MON
  Next stop on tour= CHI
  Next stop on tour= STL
  Next stop on tour= SNF
  Next stop on tour= LAX
  Next stop on tour= SND
  Next stop on tour= HOU
  Next stop on tour= ATL
  Next stop on tour= CIN
  Next stop on tour= PIT
  Next stop on tour= PHL
  Next stop on tour= NYK
 Above is complete tour, so min length tour found.
```


TRAVELING SALESPERSON PROBLEM
Lower distance between 12 cities / Sequence / Total distance: 7577
NYK
396
MON
764
CHI
294
STL
2125
SNF
405
LAX
95
SND
1521
HOU
842
ATL
454
CIN
284
PIT
305
PHL
92

BLOCK 17 Block: DYNAMIC

How to solve big and difficult problems, using dynamic programming concept, in order to obtain an ideal solution?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Dynamic
- Distance
- Route

Source:

- Book 6
- Page 809
- DINAMB

DYNAMIC PROGRAMMING - CONCEPT

Dynamic programming (DP) is a creative approach to problem solving that involves breaking down a large and difficult problem into a series of smaller, easy-to-solve problems. By solving this series of minor problems, we are able to assemble the ideal solution to the big initial problem.

A detailed discussion of this model can be found in the development of more advanced models.o find the shortest path distance through the network, we will use the following DP recursion: F (i) = min [D (i, j) + F (j)] j where F (i) is the minimum travel distance from point i to the final destination point and D (i, j) is the distance from point i to point j.In words, the minimum distance from node i to terminal node is the minimum at all reachable points over a single arc of i of the sum of the distance from i to the adjacent node plus the minimum distance from the adjacent node to the terminal node .

GOAL

Dynamic programming illustration (to see Anderson, Sweeney & Williams, an introduction to Mgr Science, 6th Ed). We have a network of 10 cities. We want to find the length of the shortest route from city 1 to city 10.

STEP1:

Here is our primitive set of ten cities, where F (i) represents the shortest distance from the route from city i to the last city: CITIES /1..10/: F

STEP2:

The derived set ROADS lists the roads that exist between cities (note: not all city.) The pairs are directly linked by a road and roads are assumed in a way.

STEP3:

The following is classic dynamic recursion programming. In words, the shortest distance from City i to City 10 is the minimum in all cities j reachable from the sum of the distance from i to j plus the minimum distance from j to city 10;

```
MODEL:
SETS:
 CITIES /1..10/: F;
!  The derived set ROADS lists the roads that exist between the cities (note: not all city
   pairs are directly linked by a road, and roads are assumed to be one way.);
 ROADS( CITIES, CITIES)/
                1,2     1,3     1,4
                2,5     2,6     2,7
                3,5     3,6     3,7
                4,5     4,6
                5,8     5,9
                6,8     6,9
                7,8     7,9
                8,10    9,10    /: D;
 ! D( i, j) is the distance from city i to j;
ENDSETS
DATA:
 ! Here are the distances that correspond to the above links;
 D      =       1       5       2
                13      12      11
                6       10      4
                12      14
                3       9
                6       5
                8       10
                5
                2;
ENDDATA
SUBMODEL MIN1:
! If you are already in City 10, then the cost to travel to City 10 is 0;
F( @SIZE( CITIES)) = 0;
! The following is the classic dynamic programming recursion. In words, the shortest distance from
  City i to City 10 is the minimum over all cities j reachable from i of the sum of the
  distance from i to j plus the minimal distance from j to City 10;
@FOR( CITIES( i)| i #LT# @SIZE( CITIES): F( i) = @MIN( ROADS( i, j): D( i, j) + F( j)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',3);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  DISTANCE (km) ROUTE VS CITY):", @NEWLINE( 1));
@TABLE(D);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution report;
@WRITEFOR( cities(i) | i #LT# 10:'  The smallest distance between  City: ', CITIES(I), '     and    City 10:',
        @FORMAT(F(I),'%3.0f'),'km',
@NEWLINE( 1));
ENDCALC
END
```

❖ DATA

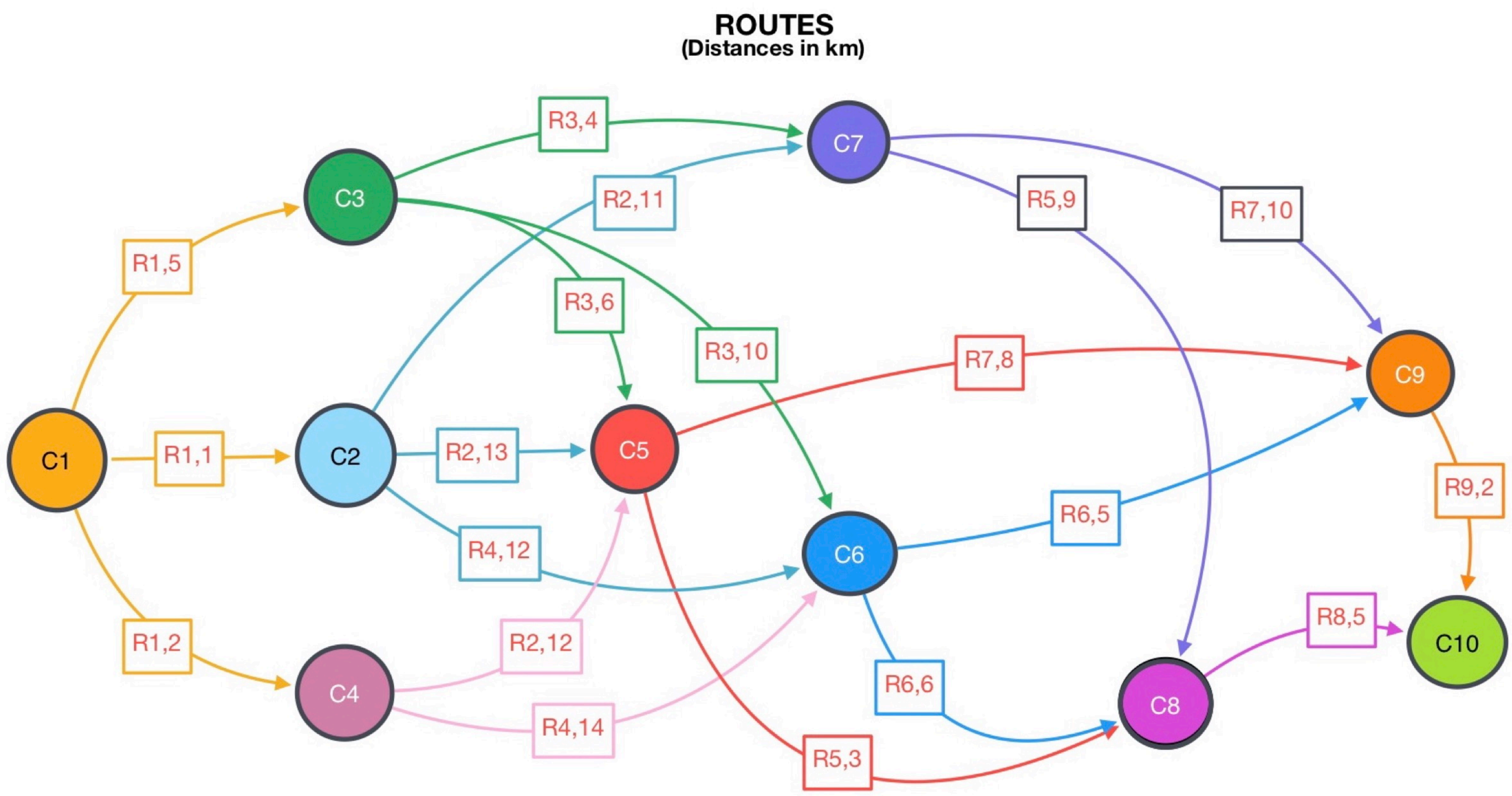
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
DISTANCE (Route vs City in km):
        1     2     3     4     5     6     7     8     9    10
 1         1.00  5.00  2.00
 2                           13.0  12.0  11.0
 3                           6.00  10.0  4.00
 4                           12.0  14.0
 5                                             3.00  9.00
 6                                             6.00  5.00
 7                                             8.00  10.0
 8                                                         5.00
 9                                                         2.00
 10
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
The smallest distance between  City: 1     and    City 10: 19km
The smallest distance between  City: 2     and    City 10: 19km
The smallest distance between  City: 3     and    City 10: 14km
The smallest distance between  City: 4     and    City 10: 20km
The smallest distance between  City: 5     and    City 10:  8km
The smallest distance between  City: 6     and    City 10:  7km
The smallest distance between  City: 7     and    City 10: 12km
The smallest distance between  City: 8     and    City 10:  5km
The smallest distance between  City: 9     and    City 10:  2km
```


ROUTES
(Distances in km)
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
R1,5
R1,1
R1,2
R3,4
R2,11
R3,6
R3,10
R2,13
R4,12
R2,12
R4,14
R7,8
R5,9
R7,10
R6,5
R6,6
R5,3
R9,2
R8,5

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Dynamic
- Distance
- Route

Source:

- Book 4
- DynamicprogShortRT

DYNAMIC PROGRAMMING - CONCEPT

Dynamic programming (DP) is a creative approach to problem solving that involves breaking down a large and difficult problem into a series of smaller, easy-to-solve problems. By solving this series of minor problems, we are able to assemble the ideal solution to the big initial problem.

A detailed discussion of this model can be found in the development of more advanced models.o find the shortest path distance through the network, we will use the following DP recursion: F (i) = min [D (i, j) + F (j)] j where F (i) is the minimum travel distance from point i to the final destination point and D (i, j) is the distance from point i to point j.In words, the minimum distance from node i to terminal node is the minimum at all reachable points over a single arc of i of the sum of the distance from i to the adjacent node plus the minimum distance from the adjacent node to the terminal node .

GOAL

Dynamic programming illustration (to see Anderson, Sweeney & Williams, an introduction to Mgt Science, 6th Ed.). We have a network of 10 cities. We want to find the length of the shortest route from city 1 to city 10.

STEP1:

Here is our primitive set of ten cities, where F (i) represents the shortest distance from the route from city i to the last city: CITIES /1..10/: F

STEP2:

The derived set ROADS lists the roads that exist between cities (note: not all city.) The pairs are directly linked by a road and roads are assumed in a way.

STEP3:

The following is classic dynamic recursion programming. In words, the shortest distance from City i to City 10 is the minimum in all cities j reachable from the sum of the distance from i to j plus the minimum distance from j to city 10;

```
MODEL:
SETS:
 ! Dynamic programming illustration (see Anderson, Sweeney & Williams, An Intro to Mgr Science, 6th Ed.).
   We have a network of 10 cities.  We want to find the length of the shortest route from city 1 to city 10.;
! Here is our primitive set of ten cities, where F( i) represents the shortest path distance from city i to the last city;
  CITIES /1..10/: F, CONPATH;
! The derived set ROADS lists the roads that exist between the cities (note: not all city pairs are directly linked by a road, and roads are
assumed to be one way.);
  ROADS( CITIES, CITIES)/
                 1,2     1,3     1,4
                 2,5     2,6     2,7
                 3,5     3,6     3,7
                 4,5     4,6
                 5,8     5,9
                 6,8     6,9
                 7,8     7,9
                 8,10            9,10/: D, RONPATH;   ! D( i, j) is the distance from city i to j;
ENDSETS
DATA:
 ! Here are the distances that correspond to the above links;
  D    =         1       5       2
                 13      12      11
                 6       10      4
                 12      14
                 3       9
                 6       5
                 8       10
                 5       2;
ENDDATA
SUBMODEL MIN2:
! If ow are already in last city, then the cost to travel to it is 0;
 F( @SIZE( CITIES)) = 0;
! The following is the classic dynamic programming recursion. In words, the shortest distance from City i to last city is the minimum
over all cities j reachable from i of the sum of the distance from i to j plus the minimal distance from j to last city;
 @FOR( CITIES( i)| i #LT# @SIZE( CITIES): F( i) = @MIN( ROADS( i, j): D( i, j) + F( j)); );
! Set CONPATH( j) > 0 if city j is on any critical path, i.e., if going to j from some i on the critical path, still allows us to get to last city in
optimal time;
 CONPATH( 1) = 1;
 @FOR( CITIES( j)| j #GT# 1: CONPATH( j) = @MAX( ROADS( i,j): CONPATH( i)*((F(I)-F( j) #EQ# D(i,j))));););
! Set RONPATH(i,j) = 1 if arc i,j is on any critical path, i.e., both i and j are on path and (i,j) is shortest path from i to j.  Note, we do not
assume distances satisfy triangle inequality;
 @FOR( ROADS(i,j): RONPATH(I,J) = @SMIN( CONPATH(i), CONPATH(j))*((F(i)-F( j)) #EQ# D(i,j));););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',3);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  DISTANCE (km) ROUTE VS CITY):", @NEWLINE( 1));
@TABLE(D);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution report;
@WRITEFOR( cities(i) | i #LT# 10:'  The smallest distance between  City: ', CITIES(I), '     and    City 10:',
        @FORMAT(F(I),'%3.0f'),'km',
@NEWLINE( 1));
ENDCALC
END
```

❖ DATA

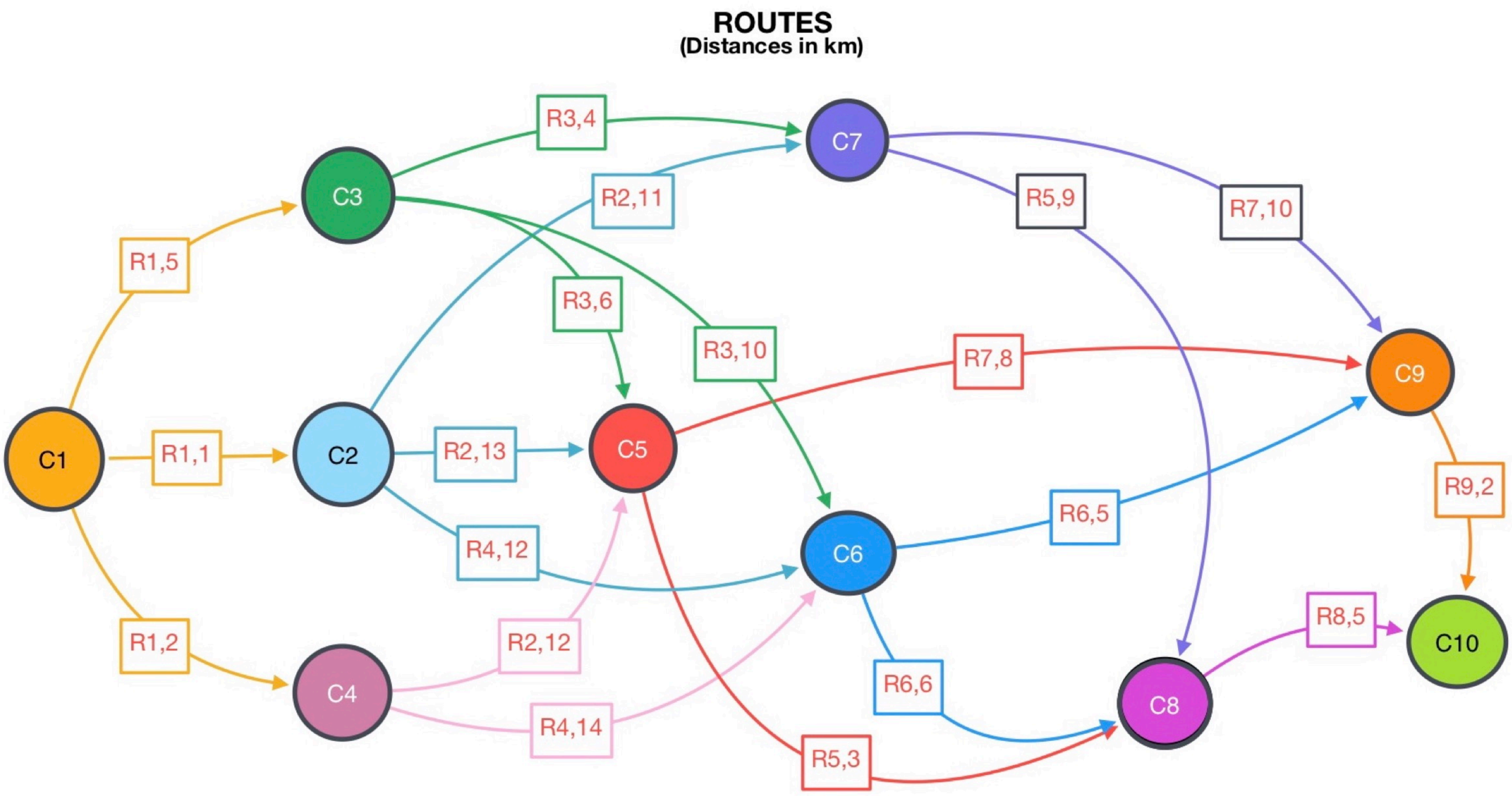
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
DISTANCE (km) ROUTE VS CITY):
        1     2     3     4     5     6     7     8     9    10
 1         1.00  5.00  2.00
 2                           13.0  12.0  11.0
 3                           6.00  10.0  4.00
 4                           12.0  14.0
 5                                             3.00  9.00
 6                                             6.00  5.00
 7                                             8.00  10.0
 8                                                         5.00
 9                                                         2.00
 10
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal solution.

```
The smallest distance between  City: 1     and    City 10: 19km
The smallest distance between  City: 2     and    City 10: 19km
The smallest distance between  City: 3     and    City 10: 14km
The smallest distance between  City: 4     and    City 10: 20km
The smallest distance between  City: 5     and    City 10:  8km
The smallest distance between  City: 6     and    City 10:  7km
The smallest distance between  City: 7     and    City 10: 12km
The smallest distance between  City: 8     and    City 10:  5km
The smallest distance between  City: 9     and    City 10:  2km
```


ROUTES
(Distances in km)
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
R1,5
R1,1
R1,2
R3,4
R2,11
R3,6
R3,10
R2,13
R4,12
R2,12
R4,14
R7,8
R5,9
R7,10
R6,5
R6,6
R5,3
R8,5
R9,2

BLOCK 18 Block: LOGISTIC

How to optimize transport resources of parts, assemblies, etc., in an industrial facility in the shortest possible time, according to available resources and thus obtain cost reduction?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Elevator
- Trip

Source:

- Book 3
- Chapter 2.5.11

GOAL

For the construction of a tower, three external elevators were installed to carry cargo.

The lifts carry one unit of load on each trip. They go up loaded and come down empty.

Elevator A caters to floors 1 and 3.

Elevator B is on floors 2 and 4.

Elevator C caters to all floors.

Each lift can perform daily, and regardless of the floor, the number of trips provided below:

	Elevator A	Elevator B	Elevator C	Demand
Floor 1	3	0	4	50
Floor 2	0	3	4	70
Floor 3	4	0	6	80
Floor 4	0	5	6	85
Available	120	95	80	-

Every day, 285 cargo units must be transported.

Since lifts operate to lubricate such equipment, elevators that serve only even or odd-numbered floors must carry out at least 10 daily trips for each floor served. The daily time available for elevator operation is 420 minutes.

Each elevator has, on each pavement served, a specific equipment of security of door, access and removal of load.

```
MODEL:
SETS:
 HEADER1 / PROD, LIMIT, VALUE /:;
  PRODUCT : AVAILABLE;
  RESOURCE: DEMAND;
  RXP( RESOURCE, PRODUCT) : TIME, PRODUCE;
  YXP( RESOURCE, HEADER1)  : SLASUR1;
  PXR( PRODUCT,    HEADER1) : SLASUR;
ENDSETS
DATA:
! Products attributes;
  PRODUCT ,          AVAILABLE          =
  ELEVATOR_A         120
  ELEVATOR_B         95
  ELEVATOR_C         80;

! Resources attributes;
  RESOURCE,          DEMAND             =
  FLOOR_1            50
  FLOOR_2            70
  FLOOR_3            80
  FLOOR_4            85;

! Required           ELEVATOR_A         ELEVATOR_B         ELEVATOR_C;
  TIME      =        3                  0                  4                  ! FLOOR_1;
                     0                  3                  4                  ! FLOOR_2;
                     4                  0                  6                  ! FLOOR_3;
                     0                  5                  6;                 ! FLOOR_4;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM( RXP(I,J ): PRODUCE( I,J));
! The available constraints;
@FOR( PRODUCT( COL):
        [AVA] @SUM( RESOURCE( LIN): PRODUCE( LIN,COL )) <= AVAILABLE( COL););
! Lifts operating time;
@FOR( PRODUCT( COL):
        [TIM] @SUM( RESOURCE( LIN): TIME( LIN,COL) *PRODUCE( LIN,COL )) <= 420;);
! Travel assistance per floor;
@FOR( RESOURCE( LIN):
        [DEM] @SUM( PRODUCT( COL): PRODUCE( LIN, COL )) = DEMAND( LIN););
! Restrictions on the minimum numbers of trip (Elevator A AND B );
@FOR(RESOURCE(LIN):
        [MNT] @SUM( PRODUCT(COL) | COL #LT# 3: PRODUCE(LIN,1)) >= 10;);
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',8);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  TIME (Min) :", @NEWLINE( 1));
@TABLE(TIME);
@WRITE(" ", @NEWLINE( 1), "  AVAILABLE (Min):", @NEWLINE( 1));
@TABLE(AVAILABLE);
@WRITE(" ", @NEWLINE( 1), "  TRIP NEED:", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J)| PRODUCE(I,J) #GT# 0: '  ',
        @FORMAT(PRODUCT( J),'-10S'), ' made:',
        @FORMAT(PRODUCE(I,J),'%5.0f'),' Trip    To    ',
        @FORMAT(RESOURCE( I),'-10s'),'  ',
@NEWLINE( 1));
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = DEMAND: ", @NEWLINE( 1));
@FOR(RXP(I,J):
        SLASUR1(I,2) = DEMAND(I);
        SLASUR1(I,1) = PRODUCE(I,1) + PRODUCE(I,2) + PRODUCE(I,3);
        SLASUR1(I,3) = SLASUR1(I,1) - SLASUR1(I,2));
@TABLE(SLASUR1);
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = MINIMUM NUMBERS OF TRIP: ", @NEWLINE( 1));
@FOR(RXP(I,J):
        SLASUR1(I,2) = PRODUCE(I,1) + PRODUCE(I,2) + PRODUCE(I,3);
        SLASUR1(I,1) = SLASUR1(I,2) - MNT(I);
        SLASUR1(I,3) = SLASUR1(I,1) - SLASUR1(I,2));
@TABLE(SLASUR1);
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = AVAILABLE: ", @NEWLINE( 1));
@FOR(PXR(I,J):
        SLASUR(I,2) = AVAILABLE(I);
        SLASUR(I,1) = SLASUR(I,2) - AVA(I) ;
        SLASUR(I,3) = SLASUR(I,1) - SLASUR(I,2));
@TABLE(SLASUR);
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT = LIFTS OPERATING TIME: ", @NEWLINE( 1));
@FOR(PXR(I,J):
        SLASUR(I,2) = AVAILABLE(I);
        SLASUR(I,1) = SLASUR(I,2) - TIM(I) ;
        SLASUR(I,3) = SLASUR(I,1) - SLASUR(I,2));
@TABLE(SLASUR);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
TIME (Min) :
            ELEVATOR_A  ELEVATOR_B  ELEVATOR_C
 FLOOR_1     3.0000000   0.0000000   4.0000000
 FLOOR_2     0.0000000   3.0000000   4.0000000
 FLOOR_3     4.0000000   0.0000000   6.0000000
 FLOOR_4     0.0000000   5.0000000   6.0000000

AVAILABLE (Min):
 ELEVATOR_A  120.00000
 ELEVATOR_B  95.000000
 ELEVATOR_C  80.000000

TRIP NEED:
 FLOOR_1     50.000000
 FLOOR_2     70.000000
 FLOOR_3     80.000000
 FLOOR_4     85.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                     285.00000
Infeasibilities:                     0.0000000

IDEAL PLANNING PROGRAM:
ELEVATOR_A   made:      5 Trip    To    FLOOR_1
ELEVATOR_C   made:     45 Trip    To    FLOOR_1
ELEVATOR_A   made:      5 Trip    To    FLOOR_2
ELEVATOR_B   made:     40 Trip    To    FLOOR_2
ELEVATOR_C   made:     25 Trip    To    FLOOR_2
ELEVATOR_A   made:     25 Trip    To    FLOOR_3
ELEVATOR_B   made:     55 Trip    To    FLOOR_3
ELEVATOR_A   made:     85 Trip    To    FLOOR_4

SLACK/SURPLUS LIMIT = DEMAND:
                    PROD        LIMIT       VALUE
 FLOOR_1       50.000000   50.000000   0.0000000
 FLOOR_2       70.000000   70.000000   0.0000000
 FLOOR_3       80.000000   80.000000   0.0000000
 FLOOR_4       85.000000   85.000000   0.0000000

SLACK/SURPLUS LIMIT = MINIMUM NUMBERS OF TRIP:
                  PROD        LIMIT        VALUE
 FLOOR_1      50.000000   50.000000    0.0000000
 FLOOR_2      70.000000   70.000000    0.0000000
 FLOOR_3      40.000000   80.000000   -40.000000
 FLOOR_4     -75.000000   85.000000   -160.00000

SLACK/SURPLUS LIMIT = AVAILABLE:
                  PROD        LIMIT        VALUE
 ELEVATOR_A   120.00000   120.00000    0.0000000
 ELEVATOR_B   95.000000   95.000000    0.0000000
 ELEVATOR_C   70.000000   80.000000   -10.000000

SLACK/SURPLUS LIMIT = LIFTS OPERATING TIME:
                  PROD        LIMIT        VALUE
 ELEVATOR_A  -185.00000   120.00000   -305.00000
 ELEVATOR_B  -205.00000   95.000000   -300.00000
 ELEVATOR_C  -60.000000   80.000000   -140.00000
```

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Plant
- Customer
- Trip

Source:

- Book 4
- CAPLOC

GOAL

Your firm has a choice of three locations to operate a manufacturing facility in. Four customers exist with a known demand for your product.

Plant / Customers		C1	C2	C3	C4	Fixed Cost	Capacity (un)
P1	$	6.00	2.00	6.00	7.00	91.00	39
P2	$	4.00	9.00	5.00	3.00	70.00	35
P3	$	8.00	8.00	1.00	5.00	24.00	31
Demand	un	15	17	22	12	-	-

Each potential plant location has an associated monthly operating cost, and shipping routes to the demand cities have varying cost.

In addition, each potential plant will have a shipping capacity that must not be exceeded.

You need to determine what plant(s) to open and how much of a product to send from each open plant to each customer to minimize total shipping cost and fixed plant operating cost.

```
MODEL:
SETS:
   PLANTS: FCOST, CAPACITY, OPEN;
   CUSTOMERS: DEMAND;
   PXC( PLANTS, CUSTOMERS)  : COST, VOL;
ENDSETS
DATA:
! The plant, their fixed cost and capacity;
   PLANTS,            FCOST,        CAPACITY =
   P1                 91            39
   P2                 70            35
   P3                 24            31;
! Customers and their demands;
   CUSTOMERS,         DEMAND =
    C1                15
    C2                17
    C3                22
    C4                12;
! The plant to cost cost/unit shipment matrix;
! REQUIRED            C1      C2      C3      C4;
 COST          =      6       2       6       7       ! P1;
                      4       9       5       3       ! P2;
                      8       8       1       5;      ! P3;
ENDDATA
SUBMODEL MIN2:
 [OBJ] MIN = @SUM( PXC: COST * VOL) + @SUM( PLANTS: FCOST * OPEN);
 ! The demand constraints;
 @FOR( CUSTOMERS( J):
          [DEM]  @SUM( PLANTS( I): VOL( I, J)) >= DEMAND( J));
! The supply constraints;
@FOR( PLANTS( I):
          [CAP]   @SUM( CUSTOMERS( J): VOL( I, J)) <= CAPACITY( I) * OPEN( I));
! Make OPEN binary(0/1);
@FOR( PLANTS: @BIN( OPEN));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Precision in digits for standard solution reports;
@SET('PRECIS',3);
! Terminal page width (0:none);
@SET('LINLEN',120);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  COST (Plant vs Customers):", @NEWLINE( 1));
@TABLE(COST);
@WRITE(" ", @NEWLINE( 1), "  FIXED COST:", @NEWLINE( 1));
@TABLE(FCOST);
@WRITE(" ", @NEWLINE( 1), "  DEMAND (un):", @NEWLINE( 1));
@TABLE(DEMAND);
@WRITE(" ", @NEWLINE( 1), "  CAPACITY (un):", @NEWLINE( 1));
@TABLE(CAPACITY);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
@SOLVE(MIN2);
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@TEXT()=@WRITEFOR( PXC(I,J) | VOL(I,J)  #GT# 0: '  From:',
           @FORMAT(PLANTS(I),'-2s'),' To:',
          @FORMAT(CUSTOMERS(J),'-2s'), ' ',
          @FORMAT(VOL(I,J),'%2.0f'), 'Un  x Unit cost: $',
          @FORMAT(COST(I,J),'%4.2f'), ' = Shipping cost:$',
          @FORMAT(VOL(I,J) * COST(I,J),'%5.2f'), ' + ', 'Fixed cost:$',
          !THE FIXED PROFIT HAD TO BE RATED TO ADJUST THE CALCULATION OF THE TOTAL PROFIT IN THE REPORT ;
          @FORMAT(@IF( PLANTS(I) #EQ# 'P1', FCOST(I)/3, FCOST(I)/2), '%4.1f'), ' = Total:$',
          @FORMAT(VOL(I,J) * COST(I,J) + @IF( PLANTS(I) #EQ# 'P1', FCOST(I)/3, FCOST(I)/2),'%6.2f'),
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN2);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
COST (Plant vs Customers):
       C1    C2    C3    C4
 P1  6.00  2.00  6.00  7.00
 P2  4.00  9.00  5.00  3.00
 P3  8.00  8.00  1.00  5.00

FIXED COST:
 P1  91.0
 P2  70.0
 P3  24.0

DEMAND (un):
 C1  15.0
 C2  17.0
 C3  22.0
 C4  12.0

CAPACITY (un):
 P1  39.0
 P2  35.0
 P3  31.0
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                                                    327.00
Objective bound:                                                                    327.00
Infeasibilities:                                                                      0.00

IDEAL PLANNING PROGRAM:
From:P1 To:C1 15Un  x Unit cost: $6.00 = Shipping cost:$90.00 + Fixed cost:$30.3 = Total:$120.33
From:P1 To:C2 17Un  x Unit cost: $2.00 = Shipping cost:$34.00 + Fixed cost:$30.3 = Total:$ 64.33
From:P1 To:C4  3Un  x Unit cost: $7.00 = Shipping cost:$21.00 + Fixed cost:$30.3 = Total:$ 51.33
From:P3 To:C3 22Un  x Unit cost: $1.00 = Shipping cost:$22.00 + Fixed cost:$12.0 = Total:$ 34.00
From:P3 To:C4  9Un  x Unit cost: $5.00 = Shipping cost:$45.00 + Fixed cost:$12.0 = Total:$ 57.00
```

3

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Plant
- Customer
- Trip

Source:

- Book 11
- Ex.: 31

GOAL

Three ships will be loaded at the port of Santos with iron ore. The Ore Terminal has four pier, each of them with a different capacity ship-loader.

Due to differences in the capacities of ships and ship-loaders, there are different loading times, depending on the combinations between ships and pier as per table below.

Terminal		Ship Loaders		
LoadTime		A	B	C
Pier	P1	7	11	9
	P2	6	10	15
	P3	12	16	14
	P4	14	6	5

Formulate the modeling so that the total time of ship loading is minimal.

```
MODEL:
SETS:
PIER: ;
SHIP: ;
RXP( PIER, SHIP): LOADTIME, PRODTIME;
ENDSETS
DATA:
! Pier attributes;
PIER    =
P1
P2
P3
P4;
! Ship attributes;
SHIP    =
A
B
C;
! Ship vs  Pier                A       B       C;
LOADTIME   =                  7       11      9       ! P1;
                              6       10      15      ! P2;
                              12      16      14      ! P3;
                              14      8       5;      ! P4;
ENDDATA
SUBMODEL  MIN3:
! Minimal  loadtime;
[OBJ] MIN = @SUM( RXP( I, J): LOADTIME( I, J) * PRODTIME( I, J));
! Pier;
@FOR( PIER( I):
        @SUM( SHIP( J): PRODTIME( I, J)) = 1);
! Ship;
@FOR( SHIP( J):
        @SUM( PIER( I): PRODTIME( I, J)) >= 1);
        @FOR(RXP(I,J): @BIN(PRODTIME(I,J)));
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE("  LOADTIME:", @NEWLINE( 1));
@TABLE(LOADTIME);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));;
! Execute sub-model;
@SOLVE(MIN3);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL LOGISTICS PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( RXP( I, J) | PRODTIME( I, J) #GT# 0: '  Pier: ', PIER( I), '   Ship: ', SHIP(J),'   Loadtime: ', 17*'.',' ',
        @FORMAT(LOADTIME(I,J),'%2.0f'),' hr',
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN3);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
LOADTIME:
              A         B         C
 P1  7.000000  11.00000  9.000000
 P2  6.000000  10.00000  15.00000
 P3  12.00000  16.00000  14.00000
 P4  14.00000  8.000000  5.000000
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              34.00000
Objective bound:                              34.00000
Infeasibilities:                              0.000000

IDEAL LOGISTICS PROGRAM:
Pier: P1   Ship: B   Loadtime: ................. 11 hr
Pier: P2   Ship: A   Loadtime: .................  6 hr
Pier: P3   Ship: A   Loadtime: ................. 12 hr
Pier: P4   Ship: C   Loadtime: .................  5 hr
```

BLOCK 19 Block: ENERGY

How to produce thermal energy for example, using Coal Mineral, Oil Diesel, etc. and respect the limits of environmental pollution allowed?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Steam
- Protection
- Blend

Source:

- Book 2
- Page 234

GOAL

The City of Criciuma operates its own electric power generation plant from coal. This plant burns three types of mineral to generate steam and thus to drive the turbines that produce electric energy.

The Department of Environmental Protection (DPA) determines that the smoke generated by burning coal can't contain more than 2500 ppm of sulfur and no more than 2.8 kg of coal dust.

The specifications of the three types of coal to be blended are shown below. The three types of coal can be mixed in any way.

Resources / Products		Coal			
		T1	T2	T3	Limit
Sulfur	ppm	1.7	3.2	2.4	2.8
Goal Powder	kg	1,000	3,500	2,700	2,500
Steam	ton	24,000	36,000	28,000	-

The plant manager wants to determine the ideal mix of generating the largest amount of steam without violating DPA determinations.

```
MODEL:
SETS:
 HEADER1 / PROD, LIMIT, VALUE /:;
 PRODUCT : STEAM, PRODUCE;
 RESOURCE: LIMIT;
 RXP( RESOURCE, PRODUCT) : USAGE;
 YXP( RESOURCE, HEADER1)  : SLASUR1;
ENDSETS
DATA:
! Resources attributes;
 RESOURCE,              LIMIT       =
 SULFUR                 2.8
 GOAL_POWDER            2500;
! Products attributes;
 PRODUCT,               STEAM       =
 COAL_T1                24000
 COAL_T2                36000
 COAL_T3                28000;
! Required              COAL_T1     COAL_T2     COAL_T3;
 USAGE          =       1.7         3.2         2.4         ! SULFOR;
                        1000        3500        2700;       ! GOAL_POWDER;
ENDDATA
SUBMODEL MIN1:
[OBJ] MIN = @SUM( PRODUCT( p): STEAM( p) * PRODUCE( p));
! Limit;
@FOR( RESOURCE( r):
      [LIM] @SUM( PRODUCT( p): USAGE( r, p) * PRODUCE( p )) >= LIMIT( r););
ENDSUBMODEL
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), "  RESOURCE (ppm,kg)):", @NEWLINE( 1));
@TABLE(USAGE);
@WRITE(" ", @NEWLINE( 1), "  LIMIT (ppm,kg):", @NEWLINE( 1));
@TABLE(LIMIT);
@WRITE(" ", @NEWLINE( 1), "  STEAM (ton):", @NEWLINE( 1));
@TABLE(STEAM);
@WRITE(" ", @NEWLINE( 1), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( PRODUCT(J)| PRODUCE(J) #GT# 0: '  Used: ', PRODUCT(J), '  ',
      @FORMAT(PRODUCE( J) * 100,'%3.1f'),'%', ' x Steam:',
      @FORMAT(STEAM(J),'%5.0f'),'ton',' = Total:',
      @FORMAT(PRODUCE( J) * STEAM( J), '%5.0f'),'ton',
@NEWLINE( 1));
! Slack/Surplus Resources report;
@WRITE(" ", @NEWLINE( 1), "  SLACK/SURPLUS LIMIT: ", @NEWLINE( 1));
@FOR(YXP(I,J):
      SLASUR1(I,2) = LIMIT(I);
      SLASUR1(I,1) = SLASUR1(I,2) - LIM(I);
      SLASUR1(I,3) = SLASUR1(I,1) - SLASUR1(I,2));
@TABLE(SLASUR1);
@WRITE(" ", @NEWLINE( 1));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:
RESOURCE (ppm,kg)):
               COAL_T1   COAL_T2   COAL_T3
 SULFOR       1.700000  3.200000  2.400000
 GOAL_POWDER  1000.000  3500.000  2700.000

LIMIT (ppm,kg):
 SULFOR       2.800000
 GOAL_POWDER  2500.000

STEAM (ton):
 COAL_T1      24000.00
 COAL_T2      36000.00
 COAL_T3      28000.00
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                              31500.00
Infeasibilities:                              0.000000

IDEAL PLANNING PROGRAM:
Used: COAL_T2  87.5% x Steam:36000ton = Total:31500ton

SLACK/SURPLUS LIMIT:
                   PROD       LIMIT       VALUE
 SULFOR        2.800000   2.800000   0.000000
 GOAL_POWDER   1937.500   2500.000  -562.5000
```

BLOCK 20

Block: ASSEMBLY LINE BALANCE

In an assembler company the big challenge is how to reduce the time in activities in the stations of works, and with that it can produce more?

OTHER AVAILABLE BLOCKS

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line Balance

INTRODUCTION

Assembly line balancing is a production strategy that sets an intended rate of production to produce a particular product within a particular time frame.

Also, the assembly line needs to be designed effectively and tasks needs to be distributed among workers, machines and work stations ensuring that every line segments in the production process can be met within the time frame and available production capacity.

Assembly line balancing can also be defined as assigning proper number of workers or machines for each operations of an assembly line so as to meet required production rate with minimum or zero ideal time.

The very purpose of line balancing is to assign workloads to each assigned work station in a manner that the every works stations has approximately same amount of work to be done.

Benefits of Assembly Line Balancing in organization.

1. Improved process efficiency
2. Increased production rate
3. Reduced total processing time
4. Minimum or Zero Ideal Time
5. Potential increase in profits and decrease in PROFITs

ASSEMBLY LINE PROBLEM

Problem:

The below product in a factory is assembled in an assembly line. This process needs to be re-arranged to find a balance that minimizes the workstation cycle time.

EXISTING ASSEMBLY LINE

Below is an assembly line showing list of 13 activities that needs to be completed to produce a product. The actual time required to produce each of this activity are as follows.

The assembly line has 5 workers (work stations) stationed on the line as follows where back tracking is not allowed.

The activities assigned to each workstation by production team are as follows.

Activity, Time: A, 30 B,50 C,40 D,50 E,20 F,10 G,10 H,20 I,10 J,30 K,20 I,50 M,10

The assembly line has 5 workers (workstations - W1, W2, W3, W4, W5) stationed on the line as follows where back tracking is not allowed.

The activities assigned to each workstation by production team are as follows.

NUMBER OF WORK STATION:

Number of Workstations (n) = 5

TOTAL PROCESSING TIME

Tp = ? Processing Time of all activities

Tp=30+50+40+50+20+20+10+10+10+20+30+50+10

Tp = 350 Seconds

CYCLE TIME OF EACH WORK STATION

Worker,	Activities,	Cycle Time
W1	A,B	80 (CW1)
W2	C,E	60 (CW2)
W3	C,E	70 (CW3)
W4	F,I,L	70 (CW4)
W5	G,J,K,M	70 (CW5)

CYCLE TIME OF ASSEMBLY LINE

Cycle time of Assembly Line is the maximum time of individual work stations.

CL = Maximum (CW1, CW2, CW3, CW4, CW5)

CL = Maximum (80, 60, 70, 70, 70)

CL = 80 Seconds

BALANCE DELAY

DL=[(5X80)–350]/[5X80]X100

DL = 12.5%

PRODUCTION RATE

Assuming Production happens 24 Hrs in 3 shifts each of 8 hrs.

Production rate (PL) = Available Time / Cycle Time PL=(24X60X60)/80

PL = 1,080 Units

Hence with existing assembly line, 1080 units can be produced per day.

REARRANGEMENT FOR BETTER BALANCE

Total Number of Workstations (n) = 5

Total Processing Time (Tp) = 350 Seconds

Average Time per work stations = Tp/ n = 350/5 = 70 Seconds

1

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Assembler
- Line
- Workstation
- Plant Floor

Source:

- Book 6
- Page 793

GOAL

This model illustrates how to balance an assembly line in order to minimize its total cycle time. An assembly line consists of a series of workstations in which each station performs one or more specialized tasks in the manufacture of a final product. The cycle time is the maximum time it takes any workstation to complete its assigned tasks.

The goal in balancing an assembly line is to assign tasks to stations, so equal amounts of work are performed at each station. Improperly balanced assembly lines will experience bottlenecks-workstations with less work are forced to wait on preceding station which have more work assigned.

The problem is complicated further by precedence relations amongst the tasks, where some tasks must be completed before others may begin (e.g., when building a computer, installing the disk drives must precede putting on the outer casing).

The assignment of tasks to workstations must obey the precedence relations.For our example, we have eleven tasks (A through K) to assign to four stations (1 through4). We need to find an assignment of tasks to workstations that minimize the assembly line's cycle time.

Given:

- A set of tasks, each with a task time,
- Precedence constraints among some of the tasks,
- A limited number of stations;

Problem:

Minimize the maximum amount of work assigned to any station, subject to:

- Each task is assigned to exactly one station,
- No task is assigned to a station prior to any of its predecessors;

```
MODEL:
! This model involves assigning tasks to stations in an assembly line so bottlenecks are avoided.
 Ideally, each station would be assigned an equal amount of work.;
SETS:
! The set of tasks to be assigned are A through K, and each task has a time to complete, T;
 TASK/ A B C D E F G H I J K/: T;
! Some predecessor, successor pairings must be observed(e.g. A must be done before B, B before C, etc.);
 PRED( TASK, TASK)/ A,B  B,C  C,F  C,G  F,J  G,J J,K  D,E  E,H  E,I  H,J  I,J /;
! There are 4 workstations;
 STATION/1..4/;
 TXS( TASK, STATION): X;
! X is the attribute from the derived set TXS that represents the assignment. X(I,K) = 1 if task I is assigned to station K;
ENDSETS
DATA:
! Data taken from Chase and Aquilano, POM. There is an estimated time required for each task:
      A  B  C  D  E  F   G   H    I  J  K;
 T = 45 11  9 50 15 12 12 12 12  8  9;
ENDDATA
SUBMODEL MIN1:
! *Warning* may be slow for more than 15 tasks. For each task, there must be one assigned station;
 @FOR( TASK( I):
       @SUM( STATION( K): X( I, K)) = 1);
! For each precedence pair, the predecessor task I cannot be assigned to a later station than its successor task J;
 @FOR( PRED( I, J):
       @SUM( STATION( K): K * X( J, K) - K * X( I, K)) >= 0);
! For each station, the total time for the assigned tasks must be less than the maximum cycle time, CYCTIME;
 @FOR( STATION( K):
       @SUM( TXS( I, K): T( I) * X( I, K)) <= CYCTIME);
! Minimize the maximum cycle time;
[OBJ] MIN = CYCTIME;
! The X(I,J) assignment variables are binary integers;
 @FOR( TXS: @BIN( X));
ENDSUBMODEL
CALC:
@SET('TERSEO',1);   ! Output level: 0=Verbose, 1-Terse;
@SET('STAWIN',0);   ! Post status windows, 1 Yes, 0 No;
! Data block;
@WRITE("  DATA:", @NEWLINE( 1), " ", @NEWLINE( 1), '  TASK:        ');
@WRITEFOR(TASK: @FORMAT(TASK, '-4s'),);
@WRITE(" ", @NEWLINE( 2), '  STATION:     ');
@WRITEFOR(STATION: @FORMAT(STATION, '-4s'),);
@WRITE(" ", @NEWLINE( 1), " ", @NEWLINE( 1),'  TIME:     ');
@WRITEFOR(TASK: @FORMAT(T, '%4.0f'),);
@WRITE(" ", @NEWLINE( 2), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN1);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( TXS( I, K) | T( I) * X( I, K) #GT# 0: '  For task:',TASK(I),', ',
       @FORMAT(T( I) * X( I, K),'%3.0f'), ' sec of work is required in station: ',
       @FORMAT(STATION(K),' 2s'),' ',
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE('  Maximum time required (sec):', 24*' ',
       @FORMAT(OBJ,'%2.0f'),
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

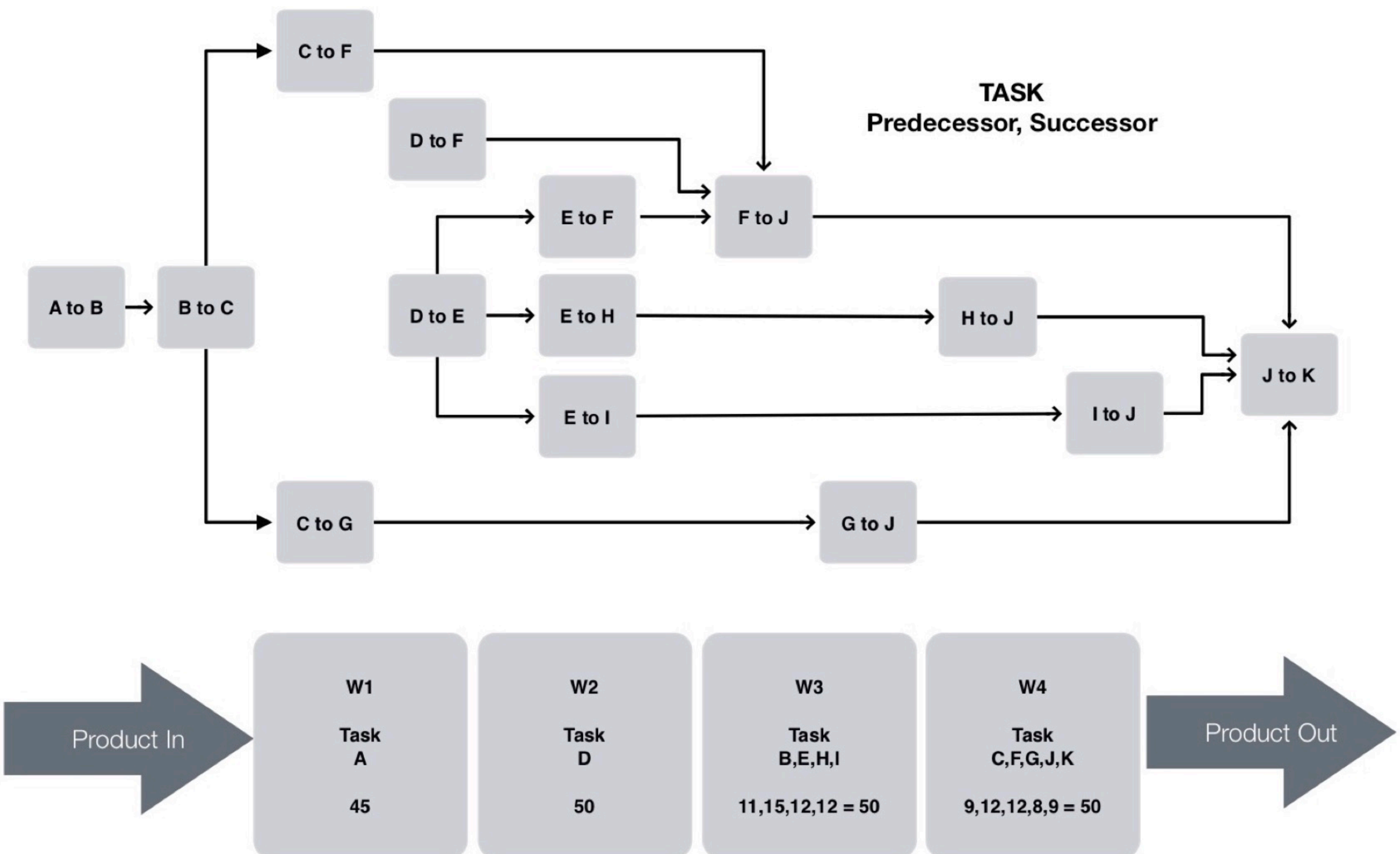
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

TASK:         A    B    C    D    E    F    G    H    I    J    K

STATION:      1    2    3    4

TIME:        45   11    9   50   15   12   12   12   12    8    9
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                      50.00000
Objective bound:                                      50.00000
Infeasibilities:                                      0.000000
IDEAL PLANNING PROGRAM:
For task:A,  45 sec of work is required in station:  1
For task:B,  11 sec of work is required in station:  3
For task:C,   9 sec of work is required in station:  4
For task:D,  50 sec of work is required in station:  2
For task:E,  15 sec of work is required in station:  3
For task:F,  12 sec of work is required in station:  4
For task:G,  12 sec of work is required in station:  4
For task:H,  12 sec of work is required in station:  3
For task:I,  12 sec of work is required in station:  3
For task:J,   8 sec of work is required in station:  4
For task:K,   9 sec of work is required in station:  4

Maximum time required (sec):                            50
```



ASSEMBLY LINE BALANCING

2

Blocks

- Product Mix
- Blend
- Finance
- Investments
- Diet
- Aviation
- Transport
- Agriculture
- Construction
- Refinery
- Schedule
- Cutting
- Metallurgy
- Fertilizer
- Clinic
- Classic
- Dynamic
- Logistics
- Energy
- Assembly Line

Keywords:

- Assembler
- Line
- Workstation
- Plant Floor

Source:

- Book 4
- ASLBALCYC

GOAL

This model illustrates how to balance an assembly line in order to minimize its total cycle time. An assembly line consists of a series of workstations in which each station performs one or more specialized tasks in the manufacture of a final product.

The cycle time is the maximum time it takes any workstation to complete its assigned tasks. The goal in balancing an assembly line is to assign tasks to stations, so equal amounts of work are performed at each station. Improperly balanced assembly lines will experience bottlenecks-workstations with less work are forced to wait on preceding station which have more work assigned.

The problem is complicated further by precedence relations amongst the tasks, where some tasks must be completed before others may begin (e.g., when building a computer, installing the disk drives must precede putting on the outer casing). The assignment of tasks to workstations must obey the precedence relations.For our example, we have eleven tasks (A through K) to assign to four stations (1 through4). We need to find an assignment of tasks to workstations that minimize the assembly line's cycle time

Given:

- A set of tasks, each with a task time,
- Precedence constraints among some of the tasks,
- A limited number of stations;

Problem:

Minimize the maximum amount of work assigned to any station, subject to:

- Each task is assigned to exactly one station,
- No task is assigned to an earlier station than any of its predecessors;

```
MODEL:
 ! Assembly line balancing model: Assign tasks to stations in an assembly line so
    1) precedence constraints among tasks are satisfied, and
    2) each station is assigned no more than a specified amount of work,
    3) Objective is to minimize the number stations required;
 ! Keywords: Line balancing, Assembly line balancing;
 SETS:
  ! There is a set of tasks, each with a duration T;
  TASK: T;
  ! Predecessor/successor pairings must be  observed(e.g. A must be done before B, B before C, etc.);
  PRED( TASK, TASK);
  ! There are a specified number of workstations;
  STATION: USE;
  TXS( TASK, STATION): X;
  !  X(i,k) = 1 if task i is assigned to station k, else 0;
 ENDSETS
 DATA:
  ! Data taken from Chase and Aquilano, POM. There is an estimated time required for each task..;
 TASK= A  B  C  D  E  F  G  H  I  J  K;
  T = 45 11  9 50 15 12 12 12 12  8  9;
 PRED= A,B  B,C  C,F  C,G  F,J  G,J J,K  D,E  E,H  E,I  H,J  I,J ;
 ! There are 5 possible stations;
 STATION= 1..5;
 ! Cycle time or upper limit on work in each station;
 CYCTIME = 50;
ENDDATA
SUBMODEL MIN2:
 ! Variables: USE(k) = 1 if station k is used, else 0, X(i,k) = 1 if task i assigned to station k, else 0;
 ! The model,   *Warning* may be slow for more than 15 tasks;
 ! Minimize the number of stations;
 [OBJ] MIN = @SUM(STATION(k): USE(k));
 ! For each task, there must be one assigned station;
 @FOR( TASK( i):
        [DOTASK] @SUM( STATION( k): X( i, k)) = 1);
 ! Precedence constraints;
 ! For each precedence pair, the predecessor task i cannot be assigned to a later station than its
  successor  task j;
 @FOR( PRED( i, j):
        [PRD] @SUM( STATION( k): k * X( j, k) - k * X( i, k)) >= 0 );
 ! For each station, the total time for the  assigned tasks must be less than the maximum cycle time, CYCTIME;
@FOR( STATION( k):
        [KNAP] @SUM( TXS( i, k): T( i) * X( i, k)) <= CYCTIME*USE(k);
        @BIN(USE(k)); ! USE(k) is a binary variable; );
 ! If station k is used, then station k-1 must also be;
  @FOR( STATION(k) | k #GT# 1: [NOGAP] USE(k) <= USE(k-1); );
 ! The X(i,j) assignment variables are binary integers;
 @FOR( TXS: @BIN( X));
 ! Some cuts;
! If task i assigned to station k, then k is used;
 @FOR( TXS(i,k):  [CUT] X(i,k) <= USE(k));
ENDSUBMODEL
```

```
CALC:
! Output level: 0=Verbose, 1-Terse;
@SET('TERSEO',1);
! Post status windows, 1 Yes, 0 No;
@SET('STAWIN',0);
! Data block;
@WRITE("  DATA:", @NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1), '  TASK:        ');
@WRITEFOR(TASK: @FORMAT(TASK, '-4s'),);
@WRITE(" ", @NEWLINE( 2), '  STATION:     ');
@WRITEFOR(STATION: @FORMAT(STATION, '-4s'),);
@WRITE(" ", @NEWLINE( 2), '  TIME:     ');
@WRITEFOR(TASK: @FORMAT(T, '%4.0f'),);
@WRITE(" ", @NEWLINE( 2), "  SOLUTION: ", @NEWLINE( 1));
! Execute sub-model;
@SOLVE(MIN2);
! Solution report;
@WRITE(" ", @NEWLINE( 1), "  IDEAL PLANNING PROGRAM: ", @NEWLINE( 1));
@WRITEFOR( TXS( I, K) | T( I) * X( I, K) #GT# 0: '  For task:',TASK(I),', ',
        @FORMAT(T( I) * X( I, K),'%3.0f'), ' sec of work is required in station: ',
        @FORMAT(STATION(K),' 2s'),' ',
@NEWLINE( 1));
@WRITE(" ", @NEWLINE( 1));
@WRITE('  Total stations required::', 27*' ',
        @FORMAT(OBJ,'%2.0f'),
@NEWLINE( 2));
!To see the corresponding model scalar, remove (!) From the line below;
!@GEN(MIN1);
ENDCALC
END
```

❖ DATA

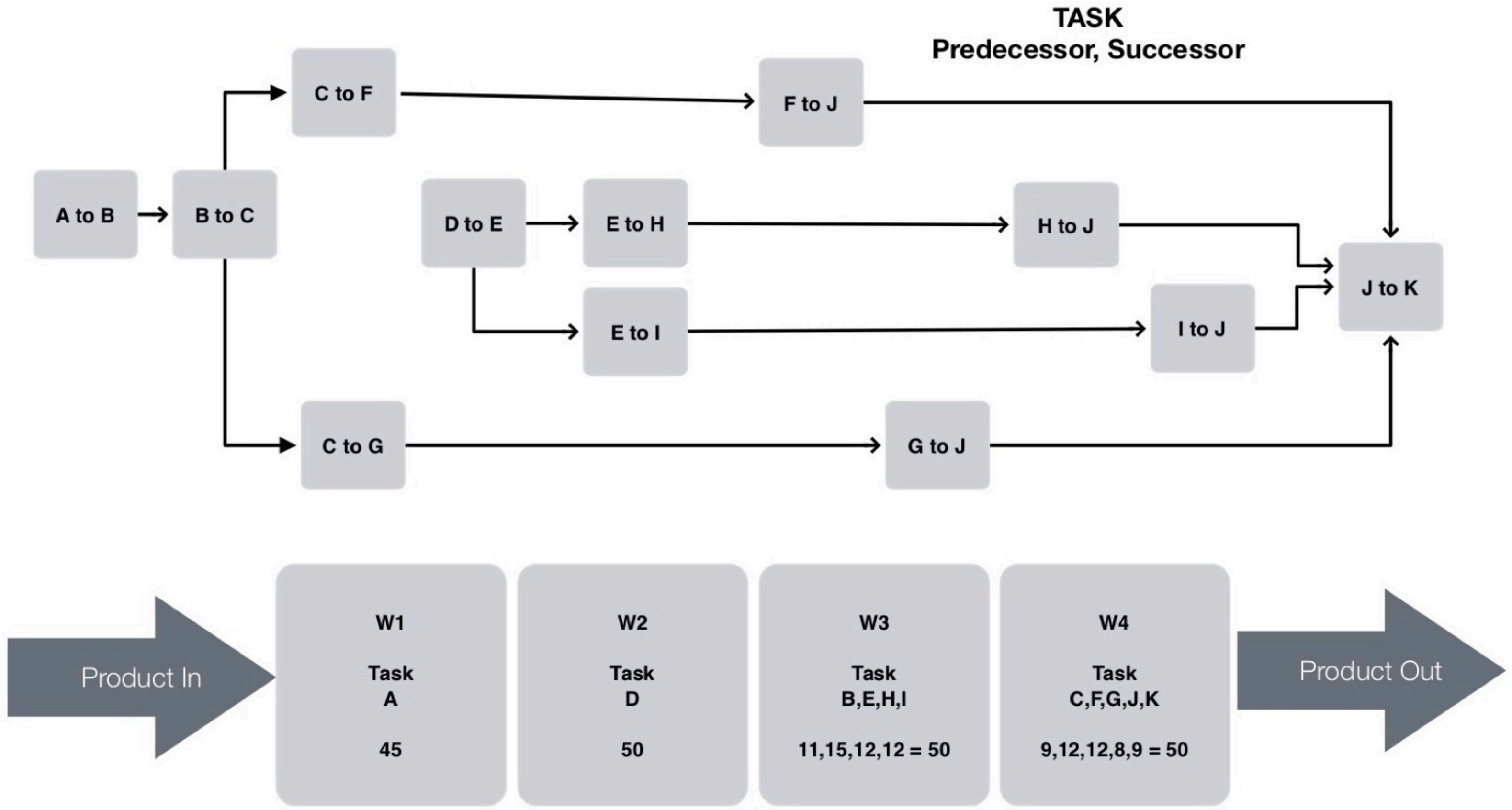
All problem data is organized in the data block as a set of members and value attributes, which can be viewed below.

```
DATA:

TASK:        A   B   C   D   E   F   G   H   I   J   K

STATION:     1   2   3   4   5

TIME:       45  11   9  50  15  12  12  12  12   8   9
```

❖ SOLUTION

Below is the solution achieved by LINGO with infeasibilities 0, and the detailed report that makes up the optimal

```
SOLUTION:
Global optimal solution found.
Objective value:                                    4.000000
Objective bound:                                    4.000000
Infeasibilities:                                    0.000000
IDEAL PLANNING PROGRAM:
For task:A,  45 sec of work is required in station:  1
For task:B,  11 sec of work is required in station:  3
For task:C,   9 sec of work is required in station:  4
For task:D,  50 sec of work is required in station:  2
For task:E,  15 sec of work is required in station:  3
For task:F,  12 sec of work is required in station:  4
For task:G,  12 sec of work is required in station:  4
For task:H,  12 sec of work is required in station:  3
For task:I,  12 sec of work is required in station:  3
For task:J,   8 sec of work is required in station:  4
For task:K,   9 sec of work is required in station:  4

Total stations required::                             4
```



ASSEMBLY LINE BALANCING

3

CHAPTER

BLOCK 1

BIBLIOGRAPHY

C3-B1 SUGGESTED BIBLIOGRAPHY FOR READING

Anderson, Sweeney, and Williams, Introduction to Management Science, 8th ed. St. Paul, MN: West Publishing, 1997.

Quantitative Methods for Business, 6th ed. St. Paul, MN: West Publishing, 1991.

Birge, J., and F. Louveaux, Introduction to Stochastic Programming, Volume 57, Springer-Verlag New York, Inc., 1997.

Black, F., and M. Scholes. (1973). "The Pricing of Options and Corporate Liabilities.", Journal of Political Economy, vol. 81, pp. 637-654.

Bradley, S.P., A.C. Hax, and T.L. Magnanti, Applied Mathematical Programming. Reading, MA : Addison-Wesley Publishing Company, Inc., 1977.

Cochran, W.G., Sampling Techniques. 2nd ed. New York, NY : Wiley, 1963.

Conway, R.W., W.L. Maxwell, and L.W. Miller, Theory of Scheduling. Reading, MA : Addison-Wesley Publishing Company, Inc.,1967.

Cox, John C. and Mark Rubinstein, Options Markets, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

Dantzig, G.B., Linear Programming and Extensions., Princeton, N.J. : Princeton University Press, 1963.

Eppen, G.D., F.J. Gould, and Schmidt, C.P., Quantitative Concepts for Management: Decision Making Without Algorithms, 3rd ed. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1989.

Introductory Management Science, 4th ed. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1993.

Gass, S., Decision Making, Models & Algorithms. New York: Wiley-Interscience, 1985.

Linear Programming, 5th ed. New York: McGraw-Hill, 1985.

Geoffrion, A., The Theory of Structured Modeling, Western Management Science Institute, UCLA, 1987.

Hadley, G., and T.M. Whitin, Analysis of Inventory Systems. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1963.

Hillier, F., and G.J. Lieberman, Introduction to Operations Research, 9th ed. New York : McGraw-Hill, Inc., 2010.

C3-B1 SUGGESTED BIBLIOGRAPHY FOR READING

Johnson, L., and D.C. Montgomery, Operations Research in Production Planning, Scheduling, and Inventory Control. New York: John Wiley & Sons, Inc., 1974.

Knowles, T., Management Science. Homewood, IL: Irwin Publishing, 1989.

Lin, S., and B. Kernighan. (1973). "An effective Heuristic Algorithm for the Traveling Salesman Problem.", Operations Research, vol. 10, pp. 463-471.

Markowitz, H. M., Portfolio Selection, Efficient Diversification of Investments, John Wiley & Sons, 1959.

Nemhauser, G., and L. Wolsey, Integer and Combinatorial Optimization, New York : John Wiley & Sons, 1988.

Moder, Joseph J., and Salah E. Elmaghraby (editors), Handbook of Operations Research, New York: Van Nostrand Reinhold Company, 1978.

Schrage, L., Optimization Modeling with LINDO, 5th. ed. Belmont, CA: Duxbury Press, 1997.

Optimization Modeling with LINGO, 6th. ed. Chicago, IL: LINDO Systems Inc, 2006.

Wagner, H.M., Principles of Management Science with Applications to Executive Decisions, 2nd ed. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1975.

Principles of Operations Research, 2nd ed. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1975.

Winston, Wayne L., Introduction to Mathematical Programming: Applications and Algorithms, 3rd ed.,. Belmont, CA: Duxbury Press, 1995.

Operations Research: Applications and Algorithms, 2nd ed., Belmont, CA: Duxbury Press, 1995

Darci Prado, Programação Linear, Serie Pesquisa Operacional Vol. 1, INDG Instituto de Desenvolvimento Gerencial S/A, 2012

Marco Cesar Goldbarg, Henrique Pacca Loureiro Luna, Elizabeth Ferreira Gouvêa Goldbarg, Programação Linear e Fluxos em Redes, Editora Campus, 2015.

Cláudio Boghi e Ricardo Shitsuka, Aplicações Práticas com o MS Office 2003 e Solver, Ferramentas Computacionais para tomada de Decisão, Editora Érica Ltda, 2005

Optimization Modeling with LINGO by Linus Schrage, www.lindo.com, Chicago

SOURCE

Book	Name	Author	Publishing company	Year
1	Programação Linear	Darci Prado	INDG	2012
2	Solver	Claudio Borghi & Ricardo Shitsuka	Editora Erica Ltda	2005
3	Programação Linear e Fluxos em Redes	Marco Cesar Goldbarg, Loureiro Lima, Elizabeth Ferreira Gouvêa Goldbarg.	Campus	2015
4	Application Model Library	Lindo Systems Inc	Lindo Systems Inc	2018
5	Optimization Modeling with LINGO	Linus Schrage	Lindo Systems Inc	2018
6	LINGO the modeling Language and Optimizar	Lindo Systems Inc	Lindo Systems Inc	2018
7	Internet Document	Msc. Fernando M. A. Nigeria	EDP/FE/UFJF	-
8	Lingo Manual de Referencias	Aloisio de Castro Gomes Junior and Marcone Jamilson Freitas Souza	UFOP	2004
9	Programação Linear	Mauricio Pereira dos Santos	DMA-IME/UFRJ	-
10	Pesquisa Operacional	Prof. Flavio Fogliato, Ph.D.	DEPROT / UFRGS	-
11	Alocação	academia.edu	-	-
12	Investigação Operacional	Manuela Magalhães Hill and Mariana Marques dos Santos	Edições Silabo Ltda	2018

Companies / Projects worked

VOLKSWAGEN

- Development, Support and IBM-APL Language Training
- Just-In-Time System (IBM-DAE)
- Broadcasting System (Siemens)
- Powertrain Warehouse Systems
- Development of tools for implementation of Regelkreis methodology in Plant Floor
- Body Paint Mix Sequencing System
- Development of Linear Programming Models using IBM-APL

GEDAS

- Just-In-Time System
- e-Procurement System

T-SYSTEMS

- e-Procurement System
- Economic Feasibility of Projects System

PROTOTYPES:

- Acquisition of Times of the activities performed during the stay of the Aircraft in soil
- Acquisition of medical data from patients with cancer, for the Genome project.
- Development of Data Analysis (BI) for various activities