```
! Setting Staff levels in a system
with random arrivals and random service times;
! Customer requests for service arrive randomly at
 one or more service groups. Each request is handled
 by one of several servers in the group. The time
 to handle the request is a random variable. If all
 servers in the group are busy, then the request
 waits until a server is free.
 For each service group we know the mean arrival rate
 of requests, the mean service time, and the standard
 deviation in service time.
  We are interested in how many servers, e.g., tech support people,
 to allocate to each service group, given a fixed
 number of servers available overall.
  Because arrivals and service times are random, if we set
 capacity only slightly larger than incoming load, there may
 be intervals in which large queues develop, and so the
 average waiting time will be large. To reduce average waiting
 time we must increase capacity above the average arriving load.
  If arrivals are in a stationary Poisson process, and
 the service times have an exponential distribution,
 the so-called M/M/c case, then the calculated results are exact,
 else they are approximate.;
! Keywords:  @PEB, Customer support, Erlang C, LINGO, M/G/C queue,
    M/M/C queue, Queuing, Service level, Staff scheduling ;
SETS:
 GROUP: ARATE, STIME, SDST,
       NSRVRS, LBSRVR, LOAD, PWAIT, WAITCND, WAITUNC, WAITMGC;
ENDSETS

DATA:
  ! Names of the service groups;
  GROUP=  TECH1 DBASE OFFICE;
  ! Arrival rate in customers/hour;
  ARATE =  1.2   2.3   2.9;
  ! Mean service time/customer in hours;
  STIME =  2.5   1.75  1.25;
  ! Standard deviation in service time;
  SDST =   2.1   2.3   1.9;
  NSAVAIL = 18; ! Number servers available overall;
ENDDATA

SUBMODEL MULTI_MGC:
! Minimize expected number waiting,
equivalently, the weighted wait time;

 MIN = @SUM( GROUP(g): ARATE(g)* WAITMGC(g));

! Cannot use more servers than available;
 @SUM( GROUP(g): NSRVRS(g)) <= NSAVAIL;

 @FOR( GROUP(g):
! Average no. of busy servers in group g;
   LOAD(g) = ARATE(g) * STIME(g);
! Lower bound on number of servers by group;
   LBSRVR(g) = @FLOOR( LOAD(g) + 0.99999);
   NSRVRS(g) >= LBSRVR(g);
   @GIN( NSRVRS(g));  ! Must be integer valued;

!  Fraction of calls that wait, the Erlang C calculation.
   Assumes Poisson arrivals, exponential service time distribution. ;
   PWAIT(g) = @PEB( LOAD, NSRVRS(g));
! Conditional expected wait, i.e., for customers who must wait,
   what is their average wait for the queue M/M/C;
   WAITCND(g) = STIME(g)/( NSRVRS(g) - LOAD);
! Unconditional expected wait, including those who wait 0;
   WAITUNC(g) = PWAIT(g) * WAITCND(g);
! An approximation for General, non-exponential service times,
```

```
  the M/G/C queue;
   WAITMGC(g) = WAITUNC(g)*(1 + SDST(g)*SDST(g)/(STIME(g)*STIME(g)))/2;
  );
ENDSUBMODEL

CALC:
 @SET("TERSEO",2); ! Turn off default output;
 @SET('GLOBAL',1); ! Use the Global solver;
 @SOLVE( MULTI_MGC);

 ! Generate a little report;
 @WRITE('   Best allocation of ',NSAVAIL,' servers.',@NEWLINE(1));
 @WRITE(' Service group  No. servers  Est. wait time', @NEWLINE(1));
 @WRITE(' -------------  -----------  --------------', @NEWLINE(1));
 @FOR( GROUP(g):
   @WRITE( @FORMAT( GROUP(g),"12s"),'      ', @FORMAT(NSRVRS(g),"5.0f"),
          '     ', @FORMAT( WAITMGC(g),"9.2f"), @NEWLINE(1));
    );
ENDCALC
```