```
SIMULATION;
OPTIONS(/ROUNDING);

comment Constants ;
INTEGER num_tellers = 3;  comment Number of tellers ;
INTEGER max_queue_size = 10;  comment Maximum queue size ;
INTEGER sim_time = 480;  comment Simulation time (in minutes) ;

comment Variables ;
INTEGER i;  comment Loop variable ;
INTEGER num_customers = 0;  comment Number of customers ;
INTEGER queue_size = 0;  comment Current queue size ;
INTEGER queue_time = 0;  comment Total time spent in queue ;
INTEGER server_busy = 0;  comment Flag indicating whether server is busy ;
INTEGER service_time = 0;  comment Time required to serve a customer ;

TABLE time_between_arrivals INTEGER(1:3);
FOR i := 1 STEP 1 UNTIL 3 DO
  time_between_arrivals(i) := i;
END;

TABLE service_times NORMAL(8, 2);

comment Random number generators ;
RANDOM int_arrival_time UNIF(time_between_arrivals);
RANDOM int_service_time NORMAL(service_times);

comment Statistics ;
TABLE wait_times HISTOGRAM(0, 30, 31);
TABLE queue_lengths HISTOGRAM(0, max_queue_size, max_queue_size+1);

comment Initialize ;
CREATE QUEUE line;
SERVER teller[num_tellers];
FOR i := 1 STEP 1 UNTIL num_tellers DO
  CREATE teller(i);

comment Main program ;
START(0);
WHILE (clock < sim_time) DO
  comment Generate new customer ;
  num_customers := num_customers + 1;
  GENERATE int_arrival_time TO line;

  comment Serve customer ;
  IF (server_busy < num_tellers) AND (queue_size = 0) THEN
    comment No waiting ;
    server_busy := server_busy + 1;
    GENERATE int_service_time TO teller(server_busy);
  ELSE
    comment Waiting in line ;
    IF (queue_size < max_queue_size) THEN
      queue_size := queue_size + 1;
      queue_time := queue_time + clock;
    END;
    comment Customer waits in line ;
    QUEUE line;
  END;

  comment Update statistics ;
```

```
  wait_times(clock - queue_time) := wait_times(clock - queue_time) + 1;
  queue_lengths(queue_size) := queue_lengths(queue_size) + 1;
END;
```