Simulation Assignment 5

Hien Ngo(431495), Iva Mamani(422759)

April 5, 2017

Exercise1

a)

The code is in APPENDIX.

b)

```
runstat(1) =  200*countervar(1) + waiting_cost   ;      % cost
runstat(2) = countervar(3)/T;      % What is the percentage of time that
    the lock is busy with operations
runstat(3) = countervar(4)/(countervar(2)+Q1+Q2);      % The average
   waiting time per ship
```

```
waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
```

- The yearly costs: 1822770.528127
- Percentage of time that the lock is busy with operation: 0.160705
- Average waiting time per ship : 0.222250

c)

```
function runstat = main
global T waiting_cost
min_cost = intmax;
runstat(1)=0;
runstat(2)=0;
    for k1=1:8
        for k2=1:8
        waiting_cost =0;
        [t, tE, x, y, Q1, Q2, eventlist, O, S, C, W]= initialization;
        % Initalize countervariable
        countervar = [O, S, C, W];
        % Perform a simulation run of one day
        while t < T   % Stopping criterium
        [t,i] = schedule_next_event(eventlist);  % Time (t) and type (i
            )
             if (i==1) || (i ==2)% arrival 1 or 2
        [x,y,Q1,Q2,eventlist,countervar] = procedure_ship_arrival(i,tE,
            t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
             elseif i ==3 % lock completion
        [x,y,Q1,Q2,eventlist,countervar] = procedure_lock_completion(tE
            ,t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
             end
```

```
        tE = t;  % move previous clock time to current time
        end
        current_cost =  200*countervar(1) + waiting_cost;
        if (min_cost >= current_cost)
            min_cost = current_cost;
            runstat(1) = k1;
            runstat(2) = k2;
        end
        end
    end
```

- Optimal value for k1: 5.
- Optimal value for k2: 3.

d)

```
% Generate interarrival time
function [a1] = arrival_realisation_type1
global T
a1=zeros;
lambda=29;
t1=-log(rand)/lambda;
I=0;
while t1<T+1
    if rand <(27+ 2*sin((t1/60)+5))
        I=I+1;
        a1(I)=t1;
    end
    t1 = t1-log(rand)/29;
end

function[a2] = arrival_realisation_type2
global T
a2=zeros;
t2=-log(rand)/25;
I=0;
while t2<T+1
    if rand <(20+ 5*sin((t2/60)+5))
        I=I+1;
        a2(I)=t2;
    end
    t2 = t2-log(rand)/25;
end
```

- We change inside of the function initialization.

```
t1 = S1(1);      %Generate first arrival of ship from the south
t2 = S2(1);      %Generate first arrival of ship from the north
```

- We change inside of the function `procedure_ship_arrival`

```
if side==1
        index_1 = index_1 + 1 ;
        eventlistn(1) =  S1(index_1);
elseif side==2
```

```
        index_2 = index_2 + 1;
        eventlistn(2) = S2(index_2);
end
```

- Optimal value for k1 changes to: 5.
- Optimal value for k2 changes to: 4.

e)

- We change some parts inside of the function `procedure_ship_arrival`

```
if y==1 % the lock opens from the south
  if side==1 %Checking for the ship from the south
      if x == 0          % the lock is idle
          if Q1 >= 10
              Q1 = Q1-10;
              countervarn(1) = countervarn(1) + 1; %update O
              countervarn(2) = countervarn(2) + 10;%update S
              service_time = service_realisation;% draw required
                  service time of arrival
              countervarn(3) = countervarn(3) + service_time;  % update
                  C
              eventlistn(3) = t+ service_time;  % next departure from
                  this server
              x = 1; % the lock becomes busy
              y=2;
              if Q1 < 10 %check if queue contains less than 10 ships
                  Q1 = Q1 +1 ;
              end
        else  % the lock is busy
          if Q1 < 10 %check if queue contains less than 10 ships
              Q1 = Q1 +1 ;
          end
        end
  elseif side==2 %ship is from north
      if Q2 < 10 %check if queue contains less than 10 ships
          Q2 = Q2 +1;
      end
  end
```

These effect decreases the yearly cost as well as the average waiting time/ship.

- The yearly costs: 1405751.34
- Average waiting time per ship : 0.121434

f)

More changes in the code can be found in APPENDIX.

```
function runstat = main
global T waiting_cost ship_rejected index_1 index_2 S1 S2
min_cost = intmax;
runstat(1)=0;
runstat(2)=0;
runstat(3)=0;
    for k1=1:8
        for k2=1:8
```

```
        index_1 = 1;
        index_2 = 1;
        S1 = arrival_realisation_type1;
        S2 = arrival_realisation_type2;
        waiting_cost =0;
        ship_rejected = 0;
        [t, tE, x, y, Q1, Q2, eventlist, O, S, C, W]= initialization;
       % Initalize countervariable
        countervar = [O, S, C, W];
       % Perform a simulation run of one day
        while t < T   % Stopping criterium
        [t,i] = schedule_next_event(eventlist);  % Time (t) and type (i
            )
            if (i==1) || (i ==2)% arrival 1 or 2
        [x,y,Q1,Q2,eventlist,countervar] = procedure_ship_arrival(i,tE,
            t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
            elseif i ==3 % lock completion
        [x,y,Q1,Q2,eventlist,countervar] = procedure_lock_completion(tE
            ,t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
            end
        tE = t;  % move previous clock time to current time
        end
        current_cost =  200*countervar(1) + waiting_cost;
        if (min_cost >= current_cost)
            min_cost = current_cost;
            runstat(1) = k1;
            runstat(2) = k2;
            runstat(3) =(ship_rejected) /(ship_rejected + countervar(2)
                +Q1+Q2+ship_rejected);
        end
        end
    end
```

- Optimal value for k1: 6
- Optimal value for k2: 5
- Percentage of arriving ships gets rejected in the optimal: 0.018077

APPENDIX

Exercise 1a)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M/M/c/c simulation in Matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Model specification M/M/c/c:
% - no waiting capacity
% - Poisson arrivals
% - Exponential service times
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Parameters:
%     T:           length of one simulation run
%
% Variables:
%     t:           current day
%     tE:          previous eventtime
%     state = [x,y,Q1,Q2] where
%         x: the state of the lock
%         y: the side at which the lock is open or at which was open the last
%    time(1=south,2=north)
%         Q1: the number of waiting ships on the south side
%         Q2: the number of waiting ships on the north side
%     eventlist = [t1,t2,c] where
%         t1 = next arrival of ship at side 1
%         t2 = next arrival of ship at side 2
%         c  = completion of a lock operation
%     countervariables = [O,S,C,W] where
%         (1) O:      number of lock operations
%         (2) S:      number of ships that went through the lock
%         (3) C:      the total time used for lock operations
%         (4) W:      the total waiting time
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mmcc_5
% Clear command screen
clc;

% Read input data
inputdata;

% Perform simulation
est = main;

% Print results
fprintf('The yearly costs: %.6f\n', est(1));
```

```
52 fprintf('Percentage of time that the lock is busy with operation:%.6f\n
       ', est(2));
53 fprintf('Average waiting time per ship : %.6f\n', est(3));
54
55
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57 % Prompt for inputdata
58 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59 function inputdata
60 global T
61 prompt  = {'Length of simulation run','Seed of random number generator'
       };
62 def     = {'365','12345'};
63 titel   = 'input';
64 lineNo  = 1;
65 parmss  = inputdlg(prompt,titel,lineNo,def);
66
67 % Check for cancel/exit
68 if( isempty(parmss) )
69         error('Input cancelled');
70 end
71 T  = str2double(parmss{1});
72 seed = str2double(parmss{2});
73
74 % Input checks
75
76 if( T <= 0 )
77         error('Simulation length must be > 0');
78 end
79
80 if( seed <= 0 )
81         error('Seed must be > 0');
82 end
83
84 % Set seed
85 rand('state',seed);  % set the seed for the random number generator
       rand()
86 randn('state',seed); % set the seed for the random number generator
       randn()
87
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 % Perform one replication
90 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91 function runstat = main
92 global T waiting_cost
93 waiting_cost =0;
94 [t, tE, x, y, Q1, Q2, eventlist, O, S, C, W]= initialization;
95 % Initalize countersdodods
96 countervar = [O, S, C, W];
97 % Perform a simulation run of one day
98     while t < T   % Stopping criterium
99     [t,i] = schedule_next_event(eventlist);  % Time (t) and type (i)
100    if (i==1) || (i ==2)% arrival 1 or 2
101        [x,y,Q1,Q2,eventlist,countervar] = procedure_ship_arrival(i,tE,
               t,x,y,Q1,Q2,eventlist,countervar);
102    elseif i ==3 % lock completion
103
```

```
104             [x,y,Q1,Q2,eventlist,countervar] = procedure_lock_completion(tE
                    ,t,x,y,Q1,Q2,eventlist,countervar);
105         end
106         tE = t;  % move previous clock time to current time
107 
108         end
109 % Compute output statistics
110 runstat(1) =  200*countervar(1) + waiting_cost  ;      % cost
111 runstat(2) = countervar(3)/T;      % What is the percentage of time that
        the lock is busy with operations
112 runstat(3) = countervar(4)/(countervar(2)+Q1+Q2);      % The average
        waiting time per ship
113 
114 
115 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
116 % Initialization function
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 function [t, tE, x, y, Q1, Q2 ,eventlist, O, S, C, W]= initialization
119 
120 t = 0.0;
121 tE = 0.0;
122 
123 x = 0;
124 y = 1;                                  %the lock opens from the south
125 Q1 =0;
126 Q2 =0;
127 
128 t1 = arrival_ship1_realisation;      %Generate first arrival of ship
        from the south
129 t2 = arrival_ship2_realisation;      %Generate first arrival of ship
        from the north
130 lock_completion = inf(1,1);          % The departure times at server
131 eventlist = [t1 ; t2 ; lock_completion];
132 O = 0.0;
133 S = 0.0;
134 C = 0.0;
135 W = 0.0;
136 
137 
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
139 % Time routine function
140 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
141 function [t,i] = schedule_next_event(eventlist)
142 [t,i] = min(eventlist); % Return time (t) and type (1/2/3)
143                         % The simulation clock t has also been updated
144 
145 
146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
147 % Arrival function
148 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149 function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_ship_arrival(
        side,tE,t,x,y,Q1,Q2,eventlist,countervar)
150 global waiting_cost
151 % Local variables
152 eventlistn = eventlist;
153 countervarn = countervar;
154 
```

```
155 % Draw interarrival time of next arrival of from the south and the
        north,
156 % and determine arrival time
157 if side==1
158     eventlistn(1) = t + arrival_ship1_realisation;
159 elseif side==2
160     eventlistn(2) = t + arrival_ship2_realisation;
161 end
162
163 countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time
164 waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
165 % Check locking condition
166 if y==1 % the lock opens from the south
167   if side==1 %Checking for the ship from the south
168       if x == 0          % the lock is idle
169           if Q1 >= 10
170               Q1 = Q1-10+1;
171               countervarn(1) = countervarn(1) + 1; %update O
172               countervarn(2) = countervarn(2) + 10;%update S
173               service_time = service_realisation;% draw required
                      service time of arrival
174               countervarn(3) = countervarn(3) + service_time;  % update
                      C
175               eventlistn(3) = t+ service_time;  % next departure from
                      this server
176               x = 1; % the lock becomes busy
177               y=2;
178           elseif (5 <= Q1) && (Q1 <=9)
179               countervarn(1) = countervarn(1) +1; %update O
180               countervarn(2) = countervarn(2) + Q1 + 1;%update S
181               service_time = service_realisation;% draw required
                      service time of arrival
182               countervarn(3) = countervarn(3) + service_time; %update C
183               eventlistn(3) = t + service_time;
184               Q1 = 0;
185               x =1 ;
186               y = 2;
187           elseif Q1 < 5
188               Q1 = Q1 +1;
189           end
190       else  % the lock is busy
191           Q1 = Q1 +1 ;
192
193       end
194   elseif side==2 %ship is from north
195           Q2 = Q2 +1;
196   end
197 elseif y==2 % the lock opens from the north
198     if side==2 %Checking for the ship from the north
199       if x == 0          % the lock is idle
200           if Q2 >= 10
201               Q2 = Q2-10+1;
202               countervarn(1) = countervarn(1) + 1; %update O
203               countervarn(2) = countervarn(2) + 10;%update S
204               service_time = service_realisation;% draw required
                      service time of arrival
205               countervarn(3) = countervarn(3) + service_time ; % update
                      C1
```

```
206                 eventlistn(3) = t+ service_time;  % next departure from
                        this server
207                 x=1;
208                 y=1;
209             elseif (5 <= Q2)&& (Q2 <=9)
210                 countervarn(1) = countervarn(1) +1; %update O;
211                 countervarn(2) = countervarn(2) + Q2 + 1;%update S
212                 service_time = service_realisation;% draw required
                        service time of arrival
213                 countervarn(3) = countervarn(3) + service_time; %update C
214                 Q2 = 0;
215                 eventlistn(3) = t + service_time;  %next departure from
                        this server
216                 x =1; %the lock becomes busy
217                 y=1;
218             elseif Q2 < 5
219                 Q2= Q2 +1;
220             end
221         else          % the lock is busy
222             Q2 = Q2 +1 ;
223         end
224     elseif side==1 %ship is from the south
225             Q1 = Q1 +1 ;
226         end
227 end
228
229 %%%%%%%%%%%%%%%%%%%%%%%%
230 % Departure function
231 %%%%%%%%%%%%%%%%%%%%%%%%
232 function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_lock_completion
        (tE,t,x,y,Q1,Q2,eventlist,countervar)
233 global waiting_cost
234 % Local variables
235 eventlistn = eventlist;
236 countervarn = countervar;
237 countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time
        in day
238 waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
239 x=0;
240
241 %checking the condition of the queue
242 if y==1    % lock opens from the south
243     if Q1 == 0
244         eventlistn(3) = inf;
245     elseif Q1 > 10
246         Q1 = Q1 - 10;
247         countervarn(1) = countervarn(1) + 1;  %update O
248         countervarn(2) = countervarn(2) + 10; %update S
249         service_time = service_realisation;
250         countervarn(3) = countervarn(3) + service_time; %update C
251         x=1; % lock is busy
252         eventlistn(3) = t + service_time;
253         y=2;
254     elseif (6 <= Q1) && (Q1<= 10)
255         countervarn(1) = countervarn(1) + 1;%update O
256         countervarn(2) = countervarn(2) + Q1 ;%update S
257         Q1 =0;
258         x=1; %the lock becomes busy
```

```
        service_time = service_realisation;
        countervarn(3) = countervarn(3) + service_time; %update C
        eventlistn(3) = t + service_time;
        y=2;
    elseif Q1 <= 5
        x=0;
        eventlistn(3) = inf;
    end
elseif y ==2 %lock opens from the north
    if Q2 == 0
        eventlistn(3) = inf;
    elseif Q2 > 10
        Q2 = Q2 - 10;
        countervarn(1) = countervarn(1) + 1; %update O
        countervarn(2) = countervarn(2) + 10; %update S
        service_time = service_realisation;
        countervarn(3) = countervarn(3) + service_time; %update C
        x=1; % lock is busy
        eventlistn(3) = t + service_time;
        y=1;
    elseif (6 <= Q2)&& (Q2<= 10)
        countervarn(1) = countervarn(1) + 1; %update O
        countervarn(2) = countervarn(2) + Q2 ;%update S
        Q2 =0;
        x=1;%lock is busy
        service_time = service_realisation;
        countervarn(3) = countervarn(3) + service_time; %update C
        eventlistn(3) = t + service_time;
        y=1;
    elseif Q2 <= 5
        x=0;
        eventlistn(3) = inf;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Library routines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate interarrival time
function [a1] = arrival_ship1_realisation
a1 =  exprnd(1/27);  % generate draw from the exponential distribution

function [a2] = arrival_ship2_realisation
a2 =  exprnd(1/20); % generate draw from the exponential distribution

% Generate service time
function [s] = service_realisation
%generate draws from the distribution of the duration of the lock function
while 0==0
    Y = rand(1,1); %r(y)=1 for 0<=y<=1  which gives us Y uniformly distributed from zero to 1.
    U = rand(1,1); %generate the U independent of Y
    if (U <= ((12*Y^2)*(1-Y))/1.778) %we find c=1.778 c=f(0.667)s
        s = Y/24;
        break;
    end
end
```

1f)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mmcc_f
% Clear command screen
clc;

% Read input data
inputdata;

% Perform simulation
est = main;

% Print results
fprintf('optimal value for k1:%.2f\n', est(1));
fprintf('optimal value for k2:%.2f\n', est(2));
fprintf('percentage of arriving ships gets rejected in the optimal:%2f\n',est(3));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prompt for inputdata
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function inputdata
global T
prompt  = {'Length of simulation run','Seed of random number generator'};
def     = {'365','12345'};
titel   = 'input';
lineNo  = 1;
parmss  = inputdlg(prompt,titel,lineNo,def);

% Check for cancel/exit
if( isempty(parmss) )
        error('Input cancelled');
end
T   = str2double(parmss{1});
seed = str2double(parmss{2});

% Input checks

if( T <= 0 )
        error('Simulation length must be > 0');
end

if( seed <= 0 )
        error('Seed must be > 0');
end

% Set seed
rand('state',seed);  % set the seed for the random number generator rand()
randn('state',seed); % set the seed for the random number generator randn()
```

```
53
54 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55 % Perform one replication
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57 function runstat = main
58 global T waiting_cost ship_rejected index_1 index_2 S1 S2
59 min_cost = intmax;
60 runstat(1)=0;
61 runstat(2)=0;
62 runstat(3)=0;
63     for k1=1:8
64         for k2=1:8
65         index_1 = 1;
66         index_2 = 1;
67         S1 = arrival_realisation_type1;
68         S2 = arrival_realisation_type2;
69         waiting_cost =0;
70         ship_rejected = 0;
71         [t, tE, x, y, Q1, Q2, eventlist, O, S, C, W]= initialization;
72         % Initalize countersdodods
73         countervar = [O, S, C, W];
74         % Perform a simulation run of one day
75         while t < T   % Stopping criterium
76         [t,i] = schedule_next_event(eventlist);  % Time (t) and type (i
               )
77             if (i==1) || (i ==2)% arrival 1 or 2
78         [x,y,Q1,Q2,eventlist,countervar] = procedure_ship_arrival(i,tE,
               t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
79             elseif i ==3 % lock completion
80         [x,y,Q1,Q2,eventlist,countervar] = procedure_lock_completion(tE
               ,t,x,y,Q1,Q2,eventlist,countervar,k1,k2);
81             end
82         tE = t;  % move previous clock time to current time
83         end
84         current_cost =  200*countervar(1) + waiting_cost;
85         if (min_cost >= current_cost)
86             min_cost = current_cost;
87             runstat(1) = k1;
88             runstat(2) = k2;
89             runstat(3) =(ship_rejected) /(ship_rejected + countervar(2)
                   +Q1+Q2+ship_rejected);
90         end
91         end
92     end
93
94 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
95 % Initialization function
96 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
97 function [t, tE, x, y, Q1, Q2 ,eventlist, O, S, C, W]= initialization
98 global S1 S2
99 t = 0.0;
100 tE = 0.0;
101
102 x = 0;
103 y = 1;                                  %the lock opens from the south
104 Q1 =0;
105 Q2 =0;
106
```

```
107 t1 = S1(1);      %Generate first arrival of ship from the south
108 t2 = S2(1);      %Generate first arrival of ship from the north
109 lock_completion = inf(1,1);          % The departure times at server
110 eventlist = [t1 ; t2 ; lock_completion];
111 O = 0.0;
112 S = 0.0;
113 C = 0.0;
114 W = 0.0;
115
116 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
117 % Time routine function
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 function [t,i] = schedule_next_event(eventlist)
120 [t,i] = min(eventlist); % Return time (t) and type (1/2/3)
121                         % The simulation clock t has also been updated
122
123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 % Arrival function
125 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
126 function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_ship_arrival(
        side,tE,t,x,y,Q1,Q2,eventlist,countervar,k1,k2)
127 global waiting_cost ship_rejected index_1 index_2 S1 S2
128 % Local variables
129 eventlistn = eventlist;
130 countervarn = countervar;
131
132 % Draw interarrival time of next arrival of from the south and the
        north,
133 % and determine arrival time
134 if side==1
135         index_1 = index_1 + 1 ;
136         eventlistn(1) =  S1(index_1);
137 elseif side==2
138         index_2 = index_2 + 1;
139         eventlistn(2) = S2(index_2);
140 end
141
142 countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time
143 waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
144 % Check locking condition
145 if y==1 % the lock opens from the south
146   if side==1 %Checking for the ship from the south
147       if x == 0          % the lock is idle
148           if Q1 >= 10
149               Q1 = Q1-10;
150               countervarn(1) = countervarn(1) + 1; %update O
151               countervarn(2) = countervarn(2) + 10;%update S
152               service_time = service_realisation;% draw required
                      service time of arrival
153               countervarn(3) = countervarn(3) + service_time;  % update
                      C
154               eventlistn(3) = t+ service_time;  % next departure from
                      this server
155               x = 1; % the lock becomes busy
156               y=2;
157               if Q1 < 10
158                   Q1 = Q1 +1 ;
159               else
```

```
160                     ship_rejected =ship_rejected+1;
161                 end
162             elseif (k1-1 <= Q1) && (Q1 <=9)
163                 countervarn(1) = countervarn(1) +1; %update O
164                 countervarn(2) = countervarn(2) + Q1 + 1;%update S
165                 service_time = service_realisation;% draw required
                        service time of arrival
166                 countervarn(3) = countervarn(3) + service_time; %update C
167                 eventlistn(3) = t + service_time;
168                 Q1 = 0;
169                 x =1 ;
170                 y = 2;
171             elseif Q1 < k1-1
172                 Q1 = Q1 +1;
173             end
174         else  % the lock is busy
175             if Q1 < 10
176                 Q1 = Q1 +1 ;
177             else
178                 ship_rejected =ship_rejected+1;
179             end
180
181         end
182     elseif side==2 %ship is from north
183             if Q2 < 10
184                 Q2 = Q2 +1 ;
185             else
186                 ship_rejected =ship_rejected+1;
187             end
188     end
189 elseif y==2 % the lock opens from the north
190       if side==2 %Checking for the ship from the north
191         if x == 0           % the lock is idle
192             if Q2 >= 10
193                 Q2 = Q2-10;
194                 countervarn(1) = countervarn(1) + 1; %update O
195                 countervarn(2) = countervarn(2) + 10;%update S
196                 service_time = service_realisation;% draw required
                        service time of arrival
197                 countervarn(3) = countervarn(3) + service_time ; % update
                        C1
198                 eventlistn(3) = t+ service_time;  % next departure from
                        this server
199                 x=1;
200                 y=1;
201                  if Q2 < 10
202                     Q2 = Q2 +1;
203                 else
204                      ship_rejected =ship_rejected+1;
205                 end
206             elseif (k2-1<= Q2)&& (Q2 <=9)
207                 countervarn(1) = countervarn(1) +1; %update O;
208                 countervarn(2) = countervarn(2) + Q2 + 1;%update S
209                 service_time = service_realisation;% draw required
                        service time of arrival
210                 countervarn(3) = countervarn(3) + service_time; %update C
211                 Q2 = 0;
```

```
                eventlistn(3) = t + service_time;  %next departure from this server
                x =1; %the lock becomes busy
                y=1;

            elseif Q2 < k2-1
                Q2= Q2 +1;
            end
        else            % the lock is busy
            if Q2 < 10
            Q2 = Q2 +1 ;
            else
                ship_rejected =ship_rejected+1;
            end
        end
    elseif side==1 %ship is from the south
            if Q1 < 10
            Q1 = Q1 +1 ;
          else
                ship_rejected =ship_rejected+1;
            end
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Departure function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_lock_completion(tE,t,x,y,Q1,Q2,eventlist,countervar,k1,k2)
global waiting_cost
% Local variables
eventlistn = eventlist;
countervarn = countervar;
countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time in day
waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
x=0;

%checking the condition of the queue
if y==1    % lock opens from the south
    if Q1 == 0
        eventlistn(3) = inf;
    elseif Q1 > 10
        Q1 = Q1 - 10;
        countervarn(1) = countervarn(1) + 1;  %update O
        countervarn(2) = countervarn(2) + 10; %update S
        service_time = service_realisation;
        countervarn(3) = countervarn(3) + service_time; %update C
        x=1; % lock is busy
        eventlistn(3) = t + service_time;
        y=2;
    elseif (k1 <= Q1) && (Q1<= 10)
        countervarn(1) = countervarn(1) + 1;%update O
        countervarn(2) = countervarn(2) + Q1 ;%update S
        Q1 =0;
        x=1; %the lock becomes busy
        service_time = service_realisation;
        countervarn(3) = countervarn(3) + service_time; %update C
```

```
          eventlistn(3) = t + service_time;
          y=2;
      elseif Q1 <= k1-1
          x=0;
          eventlistn(3) = inf;
      end
elseif y ==2 %lock opens from the north
      if Q2 == 0
          eventlistn(3) = inf;
      elseif Q2 > 10
          Q2 = Q2 - 10;
          countervarn(1) = countervarn(1) + 1; %update O
          countervarn(2) = countervarn(2) + 10; %update S
          service_time = service_realisation;
          countervarn(3) = countervarn(3) + service_time; %update C
          x=1; % lock is busy
          eventlistn(3) = t + service_time;
          y=1;
      elseif (k2 <= Q2)&& (Q2<= 10)
          countervarn(1) = countervarn(1) + 1; %update O
          countervarn(2) = countervarn(2) + Q2 ;%update S
          Q2 =0;
          x=1;%lock is busy
          service_time = service_realisation;
          countervarn(3) = countervarn(3) + service_time; %update C
          eventlistn(3) = t + service_time;
          y=1;
      elseif Q2 <= k2-1
          x=0;
          eventlistn(3) = inf;
      end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Library routines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate interarrival time
function [a1] = arrival_realisation_type1
global T
a1=zeros;
lambda=29;
t1=-log(rand)/lambda;
I=0;
while t1<T+1
      if rand <(27+ 2*sin((t1/60)+5))
          I=I+1;
          a1(I)=t1;
      end
      t1 = t1-log(rand)/29;
end

function[a2] = arrival_realisation_type2
global T
a2=zeros;
t2=-log(rand)/25;
I=0;
while t2<T+1
      if rand <(20+ 5*sin((t2/60)+5))
```

```
325             I=I+1;
326             a2(I)=t2;
327         end
328         t2 = t2-log(rand)/25;
329     end
330
331     % Generate service time
332     function [s] = service_realisation
333     %generate draws from the distribution of the duration of the lock
            function
334     while 0==0
335         Y = rand(1,1); %r(y)=1 for 0<=y<=1  which gives us Y uniformly
                distributed from zero to 1.
336         U = rand(1,1); %generate the U independent of Y
337         if (U <= ((12*Y^2)*(1-Y))/1.778) %we find c=1.778 c=f(0.667)s
338             s = Y/24;
339             break;
340         end
341     end
```