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));
fprintf('Percentage of time that the lock is busy with operation:%.6f\n', est(2));
fprintf('Average waiting time per ship : %.6f\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()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform one replication
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function runstat = main
global T waiting_cost
waiting_cost =0;
[t, tE, x, y, Q1, Q2, eventlist, O, S, C, W]= initialization;
% Initalize countersdodods
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);
    elseif i ==3 % lock completion

```

```
        [x,y,Q1,Q2,eventlist,countervar] = procedure_lock_completion(tE,t,x,y,Q1,Q2,eventlist,countervar);
    end
    tE = t;  % move previous clock time to current time

    end
% Compute output statistics
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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t, tE, x, y, Q1, Q2 ,eventlist, O, S, C, W]= initialization

t = 0.0;
tE = 0.0;

x = 0;
y = 1;                              %the lock opens from the south
Q1 =0;
Q2 =0;

t1 = arrival_ship1_realisation;     %Generate first arrival of ship from the south
t2 = arrival_ship2_realisation;     %Generate first arrival of ship from the north
lock_completion = inf(1,1);         % The departure times at server
eventlist = [t1 ; t2 ; lock_completion];
O = 0.0;
S = 0.0;
C = 0.0;
W = 0.0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Time routine function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t,i] = schedule_next_event(eventlist)
[t,i] = min(eventlist); % Return time (t) and type (1/2/3)
                        % The simulation clock t has also been updated


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Arrival function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_ship_arrival(side,tE,t,x,y,Q1,Q2,eventlist,countervar)
global waiting_cost
% Local variables
eventlistn = eventlist;
countervarn = countervar;

```

```
% Draw interarrival time of next arrival of from the south and the north,
% and determine arrival time
if side==1
    eventlistn(1) = t + arrival_ship1_realisation;
elseif side==2
    eventlistn(2) = t + arrival_ship2_realisation;
end

countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time
waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
% Check locking condition
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+1;
              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;
          elseif (5 <= Q1) && (Q1 <=9)
              countervarn(1) = countervarn(1) +1; %update O
              countervarn(2) = countervarn(2) + Q1 + 1;%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;
              Q1 = 0;
              x =1 ;
              y = 2;
          elseif Q1 < 5
              Q1 = Q1 +1;
          end
      else  % the lock is busy
          Q1 = Q1 +1 ;

      end
  elseif side==2 %ship is from north
          Q2 = Q2 +1;
  end
elseif y==2 % the lock opens from the north
    if side==2 %Checking for the ship from the north
      if x == 0           % the lock is idle
          if Q2 >= 10
              Q2 = Q2-10+1;
              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 C1
              eventlistn(3) = t+ service_time;  % next departure from this server
              x=1;
              y=1;
          elseif (5 <= Q2)&& (Q2 <=9)
              countervarn(1) = countervarn(1) +1; %update O;
              countervarn(2) = countervarn(2) + Q2 + 1;%update S
              service_time = service_realisation;% draw required service time of arrival
              countervarn(3) = countervarn(3) + service_time; %update C
              Q2 = 0;
              eventlistn(3) = t + service_time;  %next departure from this server
              x =1; %the lock becomes busy
              y=1;
          elseif Q2 < 5
              Q2= Q2 +1;
          end
      else           % the lock is busy
          Q2 = Q2 +1  ;
      end
  elseif side==1 %ship is from the south
          Q1 = Q1 +1  ;
     end
end

%%%%%%%%%%%%%%%%%%%%%%%
% Departure function
%%%%%%%%%%%%%%%%%%%%%%
function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_lock_completion(tE,t,x,y,Q1,Q2,eventlist,countervar)
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 (6 <= 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 <= 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()
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform one replication
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 countersdodods
        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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t, tE, x, y, Q1, Q2 ,eventlist, O, S, C, W]= initialization
global S1 S2
t = 0.0;
tE = 0.0;

x = 0;
y = 1;                              %the lock opens from the south
Q1 =0;
Q2 =0;

```

```
t1 = S1(1);      %Generate first arrival of ship from the south
t2 = S2(1);      %Generate first arrival of ship from the north
lock_completion = inf(1,1);         % The departure times at server
eventlist = [t1 ; t2 ; lock_completion];
O = 0.0;
S = 0.0;
C = 0.0;
W = 0.0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Time routine function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t,i] = schedule_next_event(eventlist)
[t,i] = min(eventlist); % Return time (t) and type (1/2/3)
                        % The simulation clock t has also been updated

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Arrival function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y,Q1,Q2,eventlistn,countervarn] = procedure_ship_arrival(side,tE,t,x,y,Q1,Q2,eventlist,countervar,k1,k2)
global waiting_cost ship_rejected index_1 index_2 S1 S2
% Local variables
eventlistn = eventlist;
countervarn = countervar;

% Draw interarrival time of next arrival of from the south and the north,
% and determine arrival time
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

countervarn(4) = countervarn(4) +(t-tE)*(Q1+Q2);  % update waiting time
waiting_cost = waiting_cost + (t-tE)*(Q1+Q2)*24*15;
% Check locking condition
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
                  Q1 = Q1 +1 ;
              else
```

```
                  ship_rejected =ship_rejected+1;
              end
          elseif (k1-1 <= Q1) && (Q1 <=9)
              countervarn(1) = countervarn(1) +1; %update O
              countervarn(2) = countervarn(2) + Q1 + 1;%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;
              Q1 = 0;
              x =1 ;
              y = 2;
          elseif Q1 < k1-1
              Q1 = Q1 +1;
          end
      else  % the lock is busy
          if Q1 < 10
              Q1 = Q1 +1 ;
          else
              ship_rejected =ship_rejected+1;
          end

      end
  elseif side==2 %ship is from north
          if Q2 < 10
              Q2 = Q2 +1 ;
          else
              ship_rejected =ship_rejected+1;
          end
  end
elseif y==2 % the lock opens from the north
    if side==2 %Checking for the ship from the north
      if x == 0           % the lock is idle
          if Q2 >= 10
              Q2 = Q2-10;
              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 C1
              eventlistn(3) = t+ service_time;  % next departure from this server
              x=1;
              y=1;
               if Q2 < 10
                  Q2 = Q2 +1;
              else
                  ship_rejected =ship_rejected+1;
              end
          elseif (k2-1<= Q2)&& (Q2 <=9)
              countervarn(1) = countervarn(1) +1; %update O;
              countervarn(2) = countervarn(2) + Q2 + 1;%update S
              service_time = service_realisation;% draw required service time of arrival
              countervarn(3) = countervarn(3) + service_time; %update C
              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))
```

```
        I=I+1;
        a2(I)=t2;
    end
    t2 = t2-log(rand)/25;
end

% 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
```