Faculty of Computing and Informatics (FCI)
Multimedia University
Cyberjaya

TMA1301 - Computational Methods

Assignment Report

Tutorial Session: TT6L

STUDENT ID	NAME
1211101398	POH ERN QI
1211101800	TAN JIA JIN
1211103427	LAW CHIN KEAT
1221303203	TOH EE LIN
1191201569	TOO YEE SHUEN

Table of Content

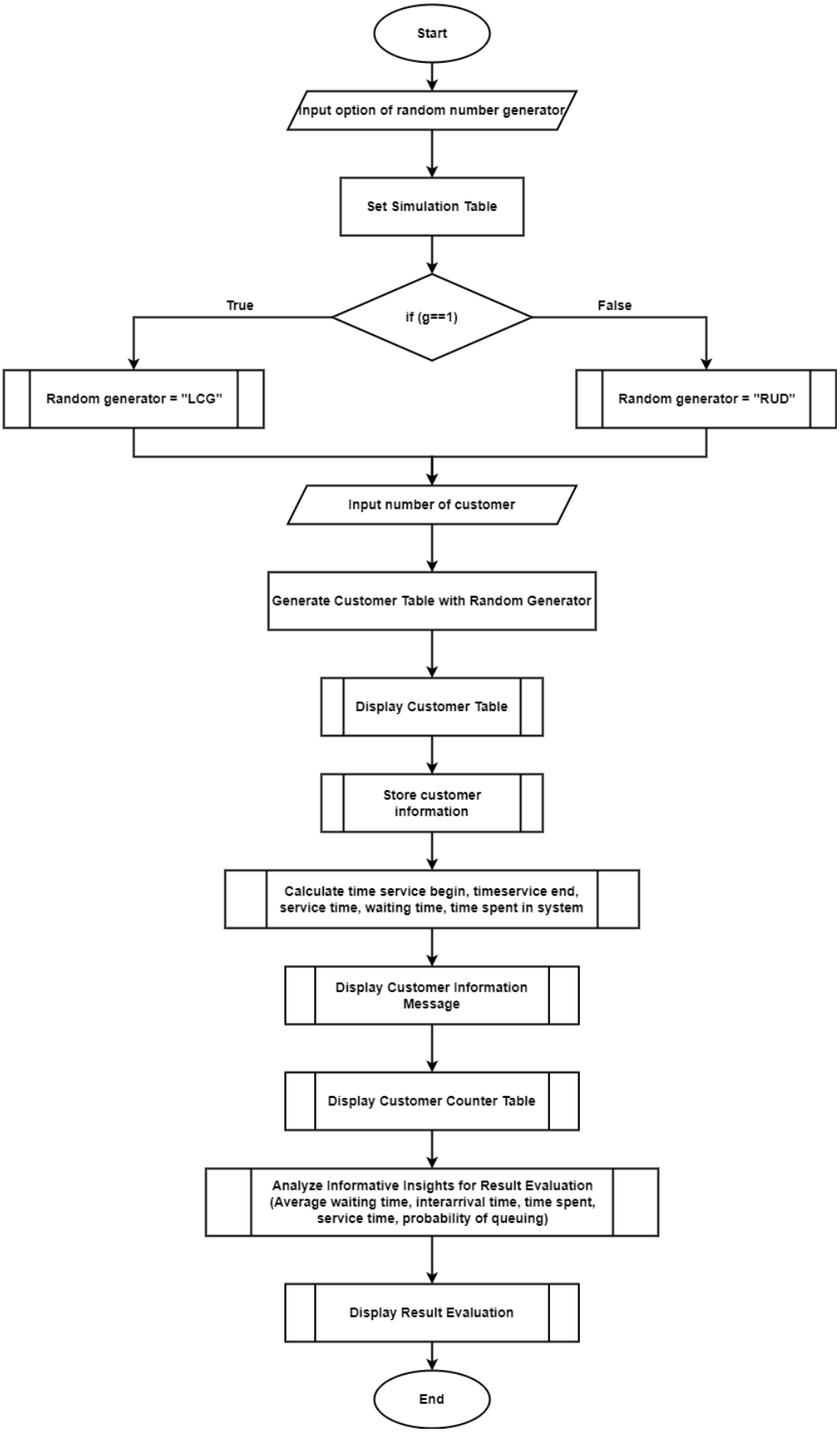
1. Details of Simulation System

The simulation system that we created is a queuing system for checkout counters for supermarkets. There are a total of four checkout counters in this simulator, where one of them is an express checkout counter and the rest of the three counters are normal checkout counters. The express checkout counter is only open for customers who buy less items.

When the user launches the simulation system, he or she needs to first choose the type of random number generator that they want to use. Then, the table of the service time for four counters and the table of the inter-arrival time for the customers will be shown by the system. After that, users will be asked to enter the number of customers. Lastly, the system will output the message for customer arrival, departure and the overall simulation table.

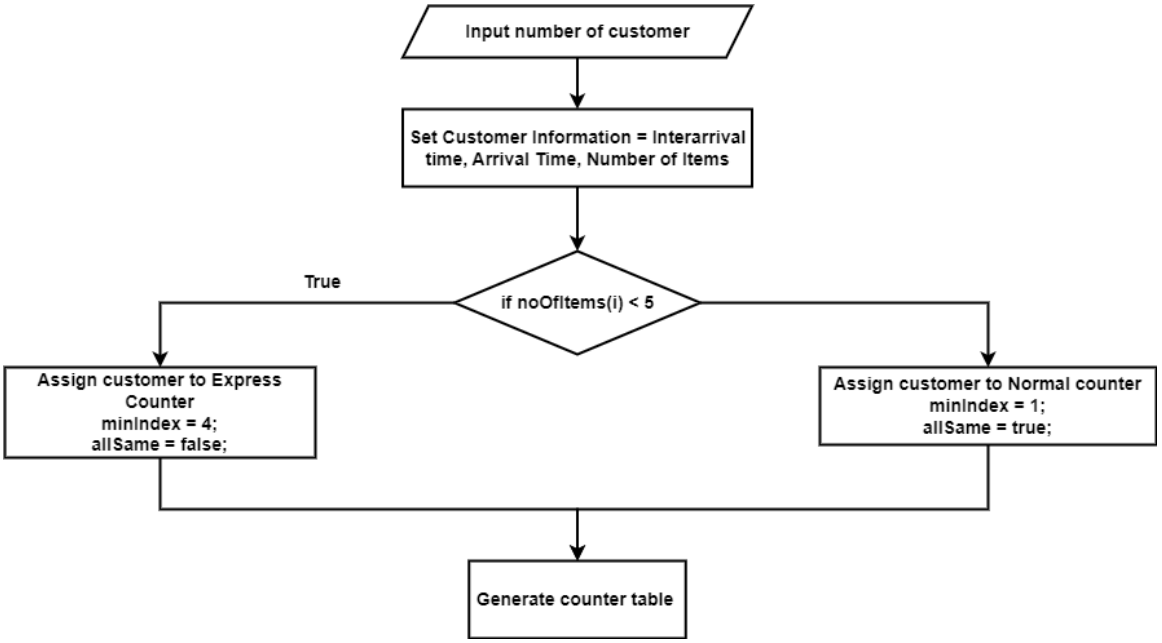
This simulation system can simulate the arrival of customers at checkout counters for supermarkets. The system will end when all the customers have arrived at the counter. The results of the simulation such as average waiting time of a customer, average inter-arrival time, average time spent, average service time for each counter and probability of a customer having to wait in a queue, will be evaluated at the end.

2. Flowchart


Start
Input option of random number generator
Set Simulation Table
if (g==1)
True
False
Random generator = "LCG"
Random generator = "RUD"
Input number of customer
Generate Customer Table with Random Generator
Display Customer Table
Store customer information
Calculate time service begin, timeservice end, service time, waiting time, time spent in system
Display Customer Information Message
Display Customer Counter Table
Analyze Informative Insights for Result Evaluation (Average waiting time, interarrival time, time spent, service time, probability of queuing)
Display Result Evaluation
End

2.1 Detail function part of the system

Customer Counter Table



3. Main Code and Algorithm of the System

Main.m

```
1   while true
2       printf('Below are the options of Random Number Generator :\n')
3       printf('1 - Linear Congruential Generators (LCG)\n2 - Random Uniform Distribution Generator (RUD)')
4       printf('\n\nPlease choose either one\n')
5       g=input('Enter : ')
6
7       if (g==1)
8           rngenerator='LCG'
9           break
10
11      elseif (g==2)
12          rngenerator='RUD'
13          break
14
15      else
16          printf('Your input is invalid! Please choose either 1 or 2\n\n')
17      end
18
19  end
```

The codes from line 1 to line 19 are to allow users to choose the type of random number generator to be used before the simulation. If the user key in input '1', Linear Congruential Generator (LCG) will be chosen as the random number generator. On the other hand, if the user key in input '2', Random Variate Generator for Uniform Distribution (RUD) will be chosen as the random number generator. A while loop is used to keep on asking for user input until the user key in either value '1' or value '2' and the system will then execute the break statement to terminate the while loop. This helps to filter out invalid inputs that may be received by the system.

```
21  %initialize every counter service time
22  counterService1 = [3, 4, 5, 6, 7];
23  counterService2 = [2, 3, 4, 5, 6];
24  counterService3 = [3, 4, 5, 6, 7];
25  counterService4 = [1, 2, 3, 4, 5];
26
27  %initialize every counter range
28  counterRange1 = [25, 45, 85, 95, 100];
29  counterRange2 = [15, 45, 70, 80, 100];
30  counterRange3 = [5, 20, 45, 75, 100];
31  counterRange4 = [25, 65, 95, 98, 100];
32
33  %put all the counter service time and range into an array
34  counterService = {counterService1, counterService2, counterService3, counterService4};
35  counterRange = {counterRange1, counterRange2, counterRange3, counterRange4};
36
37  %show all counter table
38  CounterTable(counterService, counterRange);
```

Between lines 22 and 25, we initialize the service time for each counter using four separate arrays, one for each counter.

Between lines 28 and 31, we set the range for each counter individually using four separate arrays, one for each counter. By only storing the range values, we can perform calculations to obtain the probability and cumulative distribution function (CDF).

At line 34, we have consolidated the four counter service times into a single array. Similarly, at line 35, we have combined the four counter ranges into another array.

At line 38, we just only need to pass the 2 arrays that we have created at line 34 and 35 to the CounterTable.m function and it will generate and return all of the 4 counter tables.

```
40  %initialize inter-arrival time & range
41  arrivalTime = [1, 2, 3, 4, 5, 6];
42  arrivalRange = [15, 40, 50, 70, 85, 100];
43
44  %show inter-arrival time table
45  InterArrivalTable(arrivalTime, arrivalRange);
46
47  %prompt user to enter num of customer
48  NumCustomer=input('Enter number of customer : ');
49  printf('\n');
50  CustomerCounterTable(rngenerator, NumCustomer, arrivalTime, arrivalRange, counterService, counterRange);
```

At line 41 and 42, same concept as the counter table, we stored the interarrival time and range into 1 array respectively but we do not need to store them into another array since there will be only one interarrival time and range.

At line 45, we now pass the both interarrival time and range arrays into the InterArrivalTable.m function and it will generate and return the interarrival time table.

After showing all of the counter and interarrival time tables, the program will ask for the number of customers at line 48.

Based on the input provided by the user, including the number of customers and the chosen type of random generator, the program generates the customer table.

CounterTable.m

```
1   function y = CounterTable(counterService, counterRange)
2
3       for counter = 1:numel(counterService)
4           prob = 0;
5           rangeStart = 1;
6           if (counter < 4)
7               disp(['Counter ', num2str(counter), ' (Normal)']);
8           else
9               disp(['Counter ', num2str(counter), ' (Express)']);
10          end;
11          disp('-----------------------------------------------');
12          disp('| Service Time | Probability | CDF  |  Range  |');
13          disp('-----------------------------------------------');
14
15          for i=1:numel(counterService{counter})
16              if (i == 1)
17                  prob = counterRange{counter}(i)/100;
18              else
19                  prob = counterRange{counter}(i)/100 - counterRange{counter}(i-1)/100;
20              end;
21              fprintf('|       %d      |    %.2f     | %.2f | %2d - %-3d|\n', [counterService{counter}(i), prob,
22              counterRange{counter}(i)/100, rangeStart, counterRange{counter}(i)]);
23              rangeStart = 1 + counterRange{counter}(i);
24          end
25          disp('-----------------------------------------------');
26          disp(' ');
27      end
```

In this CounterTable.m function, we used a nested for loop to display the 4 counter tables. With that, any changes we make on this nested for loop will automatically apply to four of the counter tables. The outer loop will be looped 4 times, each for each counter. While

the inner loop is to display all of the service time, probability, CDF and range for each counter.

ServiceTime.m

```
function y = ServiceTime(counterService, counterRange, rand)

    a = 1;
    b = 0;
    for i = 1:size(counterService,2)
        b = counterRange(i);
        if (rand >= a && rand <= b)
            y = counterService(i);
            break;
        end;
        a = 1 + b;
    end
```

Since every counter has different service times and each service time has a different range, it is necessary to create a function that will return the exact service time. Therefore, we have created ServiceTime.m, by passing in the counter service time, range and the random number, it will return the exact service time.

InterArrivalTable.m

```
function y = InterArrivalTable(arrivalTime, arrivalRange)
    prob = 0;
    rangeStart = 1;
    disp('Inter-arrival Time')
    disp('-----------------------------------------------------');
    disp('| Inter-arrival Time | Probability | CDF  |  Range  |');
    disp('-----------------------------------------------------');
    for i=1:size(arrivalTime,2)
        if (i == 1)
            prob = arrivalRange(i)/100;
        else
            prob = arrivalRange(i)/100 - arrivalRange(i-1)/100;
        end;
        fprintf('|         %d          |    %.2f     | %.2f | %2d - %-3d|\n', [arrivalTime(i), prob, arrivalRange(i)/100, rangeStart, arrivalRange(i)]);
        rangeStart = 1 + arrivalRange(i);
    end
    disp('-----------------------------------------------------');
```

By getting the inter arrival time and range from the Main.m, it will generate the inter arrival time table and return back to the Main.m function.

InterArrivalTime.m

```
1   function y = InterArrivalTime(arrivalTime, arrivalRange, rand)
2
3       a = 1;
4       b = 0;
5       for i = 1:size(arrivalTime,2)
6           b = arrivalRange(i);
7           if (rand >= a && rand <= b)
8               y = arrivalTime(i);
9               break;
10          end;
11          a = 1 + b;
12      end
```

Same as ServiceTime.m, by passing in the arrival time, range and the random number into this InterArrivalTime.m, it will return the exact inter arrival time.

CustomerCounterTable.m

```
1   function y = CustomerCounterTable(rngenerator, NumCustomer, InterarrivalTime, arrivalRange, counterService, counterRange)
2
3       %store the numofcustomer from 1 to n customers from input
4       NumofCustomer_arr = [1:NumCustomer];
5
6       if (rngenerator == 'LCG')
7           %generate rnInterarrival from 1 to 100
8           rnInterarrival = LCG(NumCustomer, 1, 100);
9
10          %set first rnInterarrival to 0
11          rnInterarrival(1) = 0;
12
13          rnService = LCG(NumCustomer, 1, 100);
14          noOfItems = LCG(NumCustomer, 1, 15);
15
16      else %indicates generator ==RUD
17          rnInterarrival = RUD(NumCustomer, 1, 100);
18          rnInterarrival(1) = 0;
19          rnService = RUD(NumCustomer, 1, 100);
20          noOfItems = RUD(NumCustomer, 1, 15);
21      end
```

At line 4, based on the number of customers from input, it will generate an array starting from 1 to n number of customers from input.

From line 6 to line 21, based on the chosen generator, if the chosen random generator is ‘LCG’, it will generate an array of random interarrival (rnInterarrival) by calling the function LCG by passing value from 1 to 100. After generating the random numbers, we set the first rnInterarrival to 0 for the first customer. Same goes to random service time and no.of items, we generate random service time from 1 to 100 and no.of items from 1 to 15. If the chosen random number generator is not ‘LCG’, it will go to the else statement to generate rnInterarrival, rnService, and no.of items by calling the RUD function.

```
24      InterArrTime = zeros(1, NumCustomer);
25
26      %for each rnInterarrival, generate exact interarrtime based on range
27      for i = 1:numel(rnInterarrival)
28          InterArrTime(i) = InterArrivalTime(InterarrivalTime, arrivalRange, rnInterarrival(i));
29      end
30
31      InterArrTime(1) = 0;
32      arrivalTime = zeros(1, NumCustomer);
33      arrivalTime(1) = 0;
34
35
36      disp('Customer Table')
37      disp('-------------------------------------------------------------------------------------------');
38      disp('| Cust No | RN Interarival time | Interarrival time  |  Arrival Time  |   No. of Items   |');
39      disp('-------------------------------------------------------------------------------------------');
40
41
42      %calculate arrivaltime and display values in customer table
43      for i = 1:numel(NumofCustomer_arr)
44          if (i+1 <= numel(InterArrTime))
45              arrivalTime(i+1) = arrivalTime(i) + InterArrTime(i+1);
46          end
47          fprintf('|  %3d    |        %3d          |         %d          |      %3d       |       %3d        | \n', [NumofCustomer_arr(i),
48          rnInterarrival(i), InterArrTime(i), arrivalTime(i), noOfItems(i)]);
49      end
50      disp('-------------------------------------------------------------------------------------------');
```

From line 24 to 39, an array of size based on the number of customers is created and is filled with zeros. Then, for each random number inside the array InterArrTime, it will call the InterArrivalTime function to generate the exact inter arrival time based on the range. The first inter arrival time for the first customer is set to zero. An arrivalTime array is also created and the first customer's arrival time is set to 0 as well and the customer table is displayed.

From line 42 to 50, for each customer inside the NumofCustomer_arr, as long as the next customer is less than or equal to number of elements inside the InterArrTime array, it will calculate the arrivalTime of next customer by adding the InterArrTime of next customer and arrivalTime of previous customer and print the values for each column in the table.

```
53      %cust no in each counter
54      no_cust_counter1 = zeros(1, NumCustomer);
55      no_cust_counter2 = zeros(1, NumCustomer);
56      no_cust_counter3 = zeros(1, NumCustomer);
57      no_cust_counter4 = zeros(1, NumCustomer);
58
59      %store the index of customer
60      no_cust_counter = {no_cust_counter1, no_cust_counter2, no_cust_counter3, no_cust_counter4};
61
62      %store the total number of customer in each counter
63      numcust = zeros(1, numel(no_cust_counter));
64
65      %store the total number of all customer in the queue of each counter
66      totalitemsincounter = zeros(1, numel(no_cust_counter));
67
```

To assign a customer to a counter, we first need to create an array as a queue for each counter from line 54 to 57. We initialize the queue with all zeros, and the number of zeros is the total number of customers for all counters. This is because even if we assign all customers to a counter, the maximum number of customers in that queue will be the total number of customers as well.

At line 60, we store the 4 arrays of the queue into one array. It is easier to access the array.

At line 63, this array will store the number of customers of each queue, this is used for comparison later on.

At line 66, this array will store the total number of items of the customers of each queue. It is used for comparison as well.

```
69      for i = 1:NumCustomer
70          if noOfItems(i) < 5
71              minIndex = 4;
72              allSame = true;
73
74              for j = 1:3
75                  if numcust(j) < numcust(minIndex)
76                      minIndex = j;
77                      allSame = false;
78                  elseif numcust(j) > numcust(minIndex)
79                      allSame = false;
80                  end
81              end
82
83              if allSame
84                  for k = 1:3
85                      totalitem = 0;
86                      for j = 1:numcust(k)
87                          totalitem = totalitem + noOfItems(no_cust_counter{k}(j));
88                      end
89                      totalitemsincounter(k) = totalitem;
90                  end
91
92                  minIndex = 4;
93
94                  for k = 2:3
95                      if totalitemsincounter(k) < totalitemsincounter(minIndex)
96                          minIndex = k;
97                      end
98                  end
99              end
100
101             no_cust_counter{minIndex}(numcust(minIndex)+1) = i;
```

```
101             no_cust_counter{minIndex}(numcust(minIndex)+1) = i;
102
103         else
104             minIndex = 1;
105             allSame = true;
106
107             for j = 2:3
108                 if numcust(j) < numcust(minIndex)
109                     minIndex = j;
110                     allSame = false;
111                 elseif numcust(j) > numcust(minIndex)
112                     allSame = false;
113                 end
114             end
115
116             if allSame
117                 totalitemsincounter = zeros(1, numel(no_cust_counter));
118                 for k = 1:3
119                     totalitem = 0;
120                     for j = 1:numcust(k)
121                         totalitem = totalitem + noOfItems(no_cust_counter{k}(j));
122                     end
123                     totalitemsincounter(k) = totalitem;
124                 end
125
126                 minIndex = 1;
127
128                 for k = 2:3
129                     if totalitemsincounter(k) < totalitemsincounter(minIndex)
130                         minIndex = k;
131                     end
132                 end
133             end
134
135             no_cust_counter{minIndex}(numcust(minIndex)+1) = i;
136         end
```

```
138         for j = 1:numel(no_cust_counter)
139             count = 0;
140             for k = 1:NumCustomer
141                 if no_cust_counter{j}(k) == 0
142                     break;
143                 else
144                     count = count + 1;
145                 end
146             end
147             numcust(j) = count;
148         end
149     end
```

We will be using an outer for loop to assign every customer to one of the counters. In our queue simulator, every customer will be randomly assigned 1 to 15 items. Therefore, the customer who has bought less than 5 items has the right to queue on counter 4 which is an express counter. In other words, the customer who has bought more than 5 items can only queue on counter 1 to 3.

Firstly, the customer will be assigned to the counter which has the shorter lane. This is the first preference we will look at. If there are more than 1 queue with the same number of customers and this number is the least among all of the queue, the customer will be assigned to the counter number based on ascending. So how do we code this thing out?

If the number of items of the customer is less than 5, we will initialize the counter as fourth counter, and now we will check which counter has the shorter lane since shorter lane is our first preference. If there are more than 1 counter has the same shorter lane, then the program will go to another loop from line 83 to 99. This loop will compare which queue has the lesser total number of items of the total of customers on that queue and assign the customer to it.

Else, the customer can only be assigned to counter 1 to 3. With the same concept, the program will check which counter has the shortest queue and will assign the customer to it. If there are more than 1 counter has the same shorter lane, then the program will check the total number of items of all the customers on that queue and assign the customer to the lesser number of items queue.

We will change the 0 number in the array of the counter to the index number of the customer. With this index number, we can easily know which customer is assigned to which counter.

```
151     Cus_TimeServiceBegin = zeros(1, NumCustomer);
152     Cus_TimeServiceEnds= zeros(1, NumCustomer);
153     Cus_ServiceTime=zeros(1, NumCustomer);
154     Cus_WaitingTime= zeros(1, NumCustomer);
155     Cus_TimeSpentInSystem= zeros(1, NumCustomer);
156
157     %calculate time service begin, timeservice end, service time, waiting time, time spent in system
158     for counter = 1:4
159         for customer = 1:numcust(counter)
160             if(customer==1) % if is first customer
161              %set arrivalTime, waitingTime to 0
162             Cus_TimeServiceBegin(no_cust_counter{counter}(customer)) = arrivalTime(no_cust_counter{counter}(customer));
163             Cus_WaitingTime(no_cust_counter{counter}(customer))= 0;
164
165           else
166              if(arrivalTime(no_cust_counter{counter}(customer))>= Cus_TimeServiceEnds(no_cust_counter{counter}(customer-1)))
167                   Cus_TimeServiceBegin(no_cust_counter{counter}(customer)) = arrivalTime(no_cust_counter{counter}(customer));
168              else
169                   Cus_TimeServiceBegin(no_cust_counter{counter}(customer)) = Cus_TimeServiceEnds(no_cust_counter{counter}(customer-1)) ;
170              end
171               %set waitingTime to 0 if obtain negative value using max
172              Cus_WaitingTime(no_cust_counter{counter}(customer))= max(0,Cus_TimeServiceBegin(no_cust_counter{counter}(customer)) -
173              arrivalTime(no_cust_counter{counter}(customer)));
174
175           end
176             Cus_ServiceTime(no_cust_counter{counter}(customer))= ServiceTime(counterService{counter}, counterRange{counter}, rnService(no_cust_counter{counter}(customer)));
177             Cus_TimeServiceEnds(no_cust_counter{counter}(customer)) =  Cus_ServiceTime(no_cust_counter{counter}(customer)) + Cus_TimeServiceBegin(no_cust_counter{counter}
178             (customer));
179             Cus_TimeSpentInSystem(no_cust_counter{counter}(customer))= Cus_ServiceTime(no_cust_counter{counter}(customer))+ Cus_WaitingTime(no_cust_counter{counter}(customer));
180         end
181     end
```

From line 151 to 155, arrays of Cus_TimeServiceBegin, Cus_TimeServiceEnds, Cus_ServiceTime, Cus_WaitingTime, and Cus_TimeSpentInSystem of size NumCustomer is created and filled with zeros.

From line 158 to 181, the outer for loop represents the counter from counter 1 to counter 4, the inner for loop represents customer 1 to the total number of customers in each counter. If it is the first customer in each counter, we set the Cus_TimeServiceBegin of the particular customer index to the arrival time of the customer and Cus_WaitingTime of that customer to 0. If it is not the first customer in the counter and the arrival time of that customer is larger or equal to the previous Cus_TimeServiceEnds, the Cus_TimeServiceBegin is equal to the arrivalTime of the customer. Else, the Cus_TimeServiceBegin is equal to Cus_TimeServiceEnds of the previous customer. Then, we calculate the Cus_WaitingTime by subtracting the Cus_TimeServiceBegin and the arrival time of the customer. If the calculated Cus_WaitingTime is a negative value, we use the max function in freemat to set the waiting time to 0. Finally, we generate the exact customer service time for each customer and store in array Cus_ServiceTime by calling the function ServiceTime. Lastly, we calculate the Cus_TimeServiceEnds by adding Cus_ServiceTime and Cus_TimeServiceBegin and Cus_TimeSpentInSystem by adding Cus_ServiceTime and Cus_WaitingTime.

```
186      queueCounter = zeros(1, NumCustomer);
187         for counter = 1:numel(no_cust_counter)
188             for i = 1:numcust(counter)
189             queueCounter(no_cust_counter{counter}(i)) = counter;
190         end
191     end
192
193     % Call the PrintMessage function to print the messages
194     DisplayMessage(NumCustomer, no_cust_counter, arrivalTime, queueCounter, Cus_TimeServiceEnds, Cus_TimeServiceBegin,noOfItems);
```

An array with the size of the number of customers named queueCounter is created. It then stores each customer queuing counter number in ascending order of customer index from line 187 to 191. Then it will call out the function DisplayMessage to print out the information of each customer arrival time, no. of items, time serving ends and time serving begin line by line.

```
193     %display info from counter 1 to 4
194       printf('\n');
195       for counter = 1:4
196           if (counter < 4)
197               disp(['Counter ', num2str(counter), ' (Normal)']);
198           else
199               disp(['Counter ', num2str(counter), ' (Express)']);
200           end;
201           disp('-------------------------------------------------------------------------------------------------------------------------');
202           disp('| Cust No | RN Service time | Service time | Time Service Begin | Time Service Ends | Waiting Time | Time spent in System |');
203           disp('-------------------------------------------------------------------------------------------------------------------------');
204           for i = 1:numcust(counter)
205               fprintf('|  %3d    |      %3d        |      %d       |        %3d         |       %3d         |     %3d      |          %3d         |\n', [no_cust_counter{counter}
206               (i),rnService(no_cust_counter{counter}(i)),Cus_ServiceTime(no_cust_counter{counter}(i)), Cus_TimeServiceBegin(no_cust_counter{counter}(i)),
207               Cus_TimeServiceEnds(no_cust_counter{counter}(i)),Cus_WaitingTime(no_cust_counter{counter}(i)),Cus_TimeSpentInSystem(no_cust_counter{counter}(i)) ]);
208           end
209           disp('-------------------------------------------------------------------------------------------------------------------------');
210           printf('\n');
211
212   end
213
214       %display evaluation
215     DisplayEvaluation(NumCustomer, no cust counter, numcust, Cus WaitingTime,InterArrTime, arrivalTime, Cus TimeSpentInSystem, Cus ServiceTime );
```

From line 193 to 212, the counter tables from counter 1 to counter 4 are displayed. Then, we call the DisplayEvaluation to calculate and display the evaluation of each counter and customers.

LCG.m

```
1   function y=LCG(n,min,max)
2       %X=(a*x +c)(mod m)
3
4       a=11;
5       c=29;
6       x=rand()*max;
7       x=ceil(x);
8
9       for i=1:n
10            z=a*x+c;
11          y(i)=(ceil(mod(z,max+1)));
12
13          if y(i) < min
14              y(i)=y(i)+min;
15          end
16          x=y(i);
17      end
```

The function in this LCG.m file is to generate random numbers for inter-arrival time, service time and number of items obtained by each customer using Linear Congruential Generator. The formula for Linear Congruential Generator is X = (ax+c)(mod m) and in this case, the m here is using max+1 to include the maximum value being generated. Besides, rand function is used to generate the seed number (initial value) for x. If the result of y is less than min, it will increment the result by min once.

RUD.m

```
1   function y=RUD(n,min,max)
2       % X=a+(b-a)R
3
4       r=rand(1,n);
5       k=min+(max-min)*r;
6       y=ceil(k);
```

The function in this RUD.m file is to generate random numbers for inter-arrival time, service time and number of items obtained by each customer using Random Variate Generator for Uniform Distribution. The formula for this generator is X = a+(b-a)R where a is min, b is max and R is r. The r in this case is using the rand function.

DisplayMessage.m

```
1   function y = PrintMessage(NumCustomer, counternumber, arrivalTime, queueCounter, departureTime, serviceStart,noOfItems)
2       %Print arrival, queue, and departure messages for each customer
3
4       TotalTimeUsed=departureTime(NumCustomer);
5       for i=0:TotalTimeUsed
6           for indexCus=1:NumCustomer
7               if(arrivalTime(indexCus)==i)
8                   printf('Minute %03d: Arrival of customer %d at minute %d\n',i,indexCus,arrivalTime(indexCus));
9                   printf('Minute %03d: Customer %d has %d items in hand, and Customer %d queue at the counter %d\n',i,indexCus,noOfItems(indexCus),indexCus,queueCounter(indexCus));
10              end
11              if(departureTime(indexCus)==i)
12                  printf('Minute %03d: Departure of customer %d at minute %d.\n', i,indexCus, departureTime(indexCus));
13              end
14              if(serviceStart(indexCus)==i)
15                  printf('Minute %03d: Service for customer %d started at minute %d.\n', i, indexCus, serviceStart(indexCus));
16              end
17          end
18      end
19      printf('\n');
20      %Return a success message
21      y = 'Messages printed successfully.';
22  end
```

This function in DisplayMessage is to display message for arrival, departure time and number of counter. By passing in the information, we first extract the last customer's departure time as total time used.

From line 5 to 18, the outer for loop is loop from minutes 0 to the total time used, representing each minute of the queuing process. The inner for loop is loop from 1 to the total number of customers. We will check if the arrivalTime, depatureTime and serviceStart time of current customer is equal to the current minute.Then it will print out the message for each minutes line by line.

DisplayEvaluation.m

```
1   function y = displayEvaluation(NumCustomer, no_cust_counter, numcust, Cus_WaitingTime, InterArrTime, arrivalTime,  Cus_TimeSpentInSystem, Cus_ServiceTime)
2       totalWaitTime=0;
3       totalInterArrTime=0;
4       totalArrivalTime=0;
5       totalCusTimeSpent=0;
6       count=0;
7       totalService= zeros(1, 4);
8       avgService= zeros(1,4);
9
10
11      %find totalwaitTime, totalInterArrTime, totalArrivalTime, totalCusTimeSpent
12       for customer = 1: NumCustomer
13              totalWaitTime= Cus_WaitingTime(customer)+ totalWaitTime;
14              totalInterArrTime= InterArrTime(customer)+ totalInterArrTime;
15              totalArrivalTime= arrivalTime(customer) + totalArrivalTime;
16              totalCusTimeSpent= Cus_TimeSpentInSystem(customer) +  totalCusTimeSpent;
17
18              if(Cus_WaitingTime(customer) > 0)
19                 count = count + 1;
20              end
21
22       end
```

From line 1 to 8, we first initialise the totalWaitTime, totalInterArrTime, totalArrivalTime, totalCusTimeSpent and count to 0 and create arrays of totalService and avgService of size 4 and filled it with zeros.

From line 12 to 22, for each customer till the number of customers input by the user, we calculate the total waiting time by adding the Cus_WaitingTime of each customer in the array, same concept applies to totalInterArrTime, totalArrivalTime and totalCusTimeSpent. Then, to calculate the probability of customers having to wait, we first count the number of Cus_WaitingTime that is larger than 0.

```
25      %calculate total service time for each counter
26      for counter=1:4
27           totalServiceTime=0;
28           for i = 1: numcust(counter)
29               totalServiceTime= Cus_ServiceTime(no_cust_counter{counter}(i))+ totalServiceTime;
30           end
31           totalService(counter)=totalServiceTime;
32      end
33
34
35      %find the avg and probability
36      avgWait= totalWaitTime / NumCustomer;
37      avgInterArr=  totalInterArrTime / (NumCustomer-1);
38      avgArrivalTime =  totalArrivalTime / NumCustomer;
39      avgCusTimeSpent=  totalCusTimeSpent / NumCustomer;
40      probWait= count / NumCustomer;
41
42
43      %display the message
44     fprintf('The average waiting time:  %0.4f\n\n', avgWait);
45     fprintf('The average interarrival time spent:  %0.4f\n\n', avgInterArr);
46     fprintf('The average time spent in system:  %0.4f\n\n', avgCusTimeSpent);
47
48     for i=1: numel(avgService)
49          avgService(i)= totalService(i) / numcust(i);
50          fprintf('The average service time for counter %d:  %0.4f\n\n', i, avgService(i));
51     end
52
53    fprintf('The probability of a customer having to wait:  %0.4f\n\n', probWait);
```

From line 26 to 32, we use a for loop to calculate the total service time for each counter, the outer loop is the counter from counter 1 to 4, inner loop starting from customer 1 to the total number of customers in each counter. Then we store the total service time of each counter in array called totalService.

From line 36 to 40, we find the average waiting time, average interarrival time, average arrival time, average customer time spent in system and probability of waiting time. For the case of average inter arrival time, we divided by number of customer minus 1.

From line 43 to 53, the message of average waiting time, average interarrival time, average time spent in system are displayed. Inside the for loop, we calculate the average service time for each counter and store it in array avgService and print the message. The messages are all printed in 4 decimal places.

4. Result Outputs

The output provided is a simulation of a queuing system with four counters. The simulation employs two distinct types of random number generators: the Linear Congruential Generator (LCG) and the Random Uniform Distribution Generator (RUD). These generators play a crucial role in generating random numbers that determine various aspects of the queuing system. By utilizing LCG and RUD, the simulation ensures the incorporation of realistic randomness into components such as customer arrivals, service times, interarrival times, and other critical parameters. This meticulous approach enhances the fidelity of the simulation, enabling a more accurate representation of real-world queuing scenarios.

4.1 Random Number Generators

4.1.1 Case Linear Congruential Generator (LCG) Input

```
--> Main
Below are the options of Random Number Generator :
1 - Linear Congruential Generators (LCG)
2 - Random Uniform Distribution Generator (RUD)

Please choose either one
Enter : 1

g =
 1
```

Figure 4.1.1.1 User input with option LCG

4.1.2 Case Random Uniform Distribution Generator (RUD) Input

```
--> Main
Below are the options of Random Number Generator :
1 - Linear Congruential Generators (LCG)
2 - Random Uniform Distribution Generator (RUD)

Please choose either one
Enter : 2

g =
 2
```

Figure 4.1.2.1 User input with option RUD

4.2 Simulation table

```
rngenerator =
LCG
Counter 1 (Normal)
-----------------------------------------------
| Service Time | Probability | CDF  |  Range  |
-----------------------------------------------
|       3      |    0.25     | 0.25 |  1 - 25 |
|       4      |    0.20     | 0.45 | 26 - 45 |
|       5      |    0.40     | 0.85 | 46 - 85 |
|       6      |    0.10     | 0.95 | 86 - 95 |
|       7      |    0.05     | 1.00 | 96 - 100|
-----------------------------------------------

Counter 2 (Normal)
-----------------------------------------------
| Service Time | Probability | CDF  |  Range  |
-----------------------------------------------
|       2      |    0.15     | 0.15 |  1 - 15 |
|       3      |    0.30     | 0.45 | 16 - 45 |
|       4      |    0.25     | 0.70 | 46 - 70 |
|       5      |    0.10     | 0.80 | 71 - 80 |
|       6      |    0.20     | 1.00 | 81 - 100|
-----------------------------------------------

Counter 3 (Normal)
-----------------------------------------------
| Service Time | Probability | CDF  |  Range  |
-----------------------------------------------
|       3      |    0.05     | 0.05 |  1 - 5  |
|       4      |    0.15     | 0.20 |  6 - 20 |
|       5      |    0.25     | 0.45 | 21 - 45 |
|       6      |    0.30     | 0.75 | 46 - 75 |
|       7      |    0.25     | 1.00 | 76 - 100|
-----------------------------------------------

Counter 4 (Express)
-----------------------------------------------
| Service Time | Probability | CDF  |  Range  |
-----------------------------------------------
|       1      |    0.25     | 0.25 |  1 - 25 |
|       2      |    0.40     | 0.65 | 26 - 65 |
|       3      |    0.30     | 0.95 | 66 - 95 |
|       4      |    0.03     | 0.98 | 96 - 98 |
|       5      |    0.02     | 1.00 | 99 - 100|
-----------------------------------------------

Inter-arrival Time
-------------------------------------------------------
| Inter-arrival Time | Probability | CDF  |  Range  |
-------------------------------------------------------
|         1          |    0.15     | 0.15 |  1 - 15 |
|         2          |    0.25     | 0.40 | 16 - 40 |
|         3          |    0.10     | 0.50 | 41 - 50 |
|         4          |    0.20     | 0.70 | 51 - 70 |
|         5          |    0.15     | 0.85 | 71 - 85 |
|         6          |    0.15     | 1.00 | 86 - 100|
-------------------------------------------------------
Enter number of customer : 20
```

Figure 4.2.1 Simulation Table specified range of attributes

Simulation table of the service time for 4 servers and inter-arrival time range for the customers. This table provides the distribution range for the service times at each counter and the inter-arrival times between customers in the queuing system. These distributions define the randomness in the system and are used to simulate customer arrivals and service times.

4.3 Customer Table

```
Enter number of customer : 20

Customer Table
```

Cust No	RN Interarival time	Interarrival time	Arrival Time	No. of Items
1	0	0	0	4
2	18	2	2	9
3	25	2	4	1
4	1	1	5	8
5	40	2	7	5
6	65	4	11	4
7	37	2	13	9
8	32	2	15	1
9	78	5	20	8
10	79	5	25	5
11	90	6	31	4
12	9	1	32	9
13	27	2	34	1
14	23	2	36	8
15	80	5	41	5
16	1	1	42	4
17	40	2	44	9
18	65	4	48	1
19	37	2	50	8
20	32	2	52	5

Figure 4.3.1 Customer Information Table with random inputs

By using the selected random number generator, the "Customer Table" will be generated randomly for displaying information about each customer, including their customer number, random number for interarrival time, actual interarrival time, arrival time, and the number of items they have.

4.4 Display messages

```
Minute 000: Arrival of customer 1 at minute 0
Minute 000: Customer 1 has 4 items in hand, and Customer 1 queue at the counter 4
Minute 000: Service for customer 1 started at minute 0.

Minute 002: Arrival of customer 2 at minute 2
Minute 002: Customer 2 has 9 items in hand, and Customer 2 queue at the counter 1
Minute 002: Service for customer 2 started at minute 2.
Minute 003: Departure of customer 1 at minute 3.

Minute 004: Arrival of customer 3 at minute 4
Minute 004: Customer 3 has 1 items in hand, and Customer 3 queue at the counter 2
Minute 004: Service for customer 3 started at minute 4.

Minute 005: Arrival of customer 4 at minute 5
Minute 005: Customer 4 has 8 items in hand, and Customer 4 queue at the counter 3
Minute 005: Service for customer 4 started at minute 5.
Minute 007: Departure of customer 2 at minute 7.

Minute 007: Arrival of customer 5 at minute 7
Minute 007: Customer 5 has 5 items in hand, and Customer 5 queue at the counter 2
Minute 008: Departure of customer 3 at minute 8.
Minute 008: Service for customer 5 started at minute 8.
Minute 011: Departure of customer 5 at minute 11.

Minute 011: Arrival of customer 6 at minute 11
Minute 011: Customer 6 has 4 items in hand, and Customer 6 queue at the counter 4
Minute 011: Service for customer 6 started at minute 11.
Minute 012: Departure of customer 4 at minute 12.

Minute 013: Arrival of customer 7 at minute 13
Minute 013: Customer 7 has 9 items in hand, and Customer 7 queue at the counter 1
Minute 013: Service for customer 7 started at minute 13.
Minute 014: Departure of customer 6 at minute 14.

Minute 015: Arrival of customer 8 at minute 15
Minute 015: Customer 8 has 1 items in hand, and Customer 8 queue at the counter 3
Minute 015: Service for customer 8 started at minute 15.
Minute 018: Departure of customer 7 at minute 18.
```

```
Minute 020: Arrival of customer 9 at minute 20
Minute 020: Customer 9 has 8 items in hand, and Customer 9 queue at the counter 2
Minute 020: Service for customer 9 started at minute 20.
Minute 021: Departure of customer 8 at minute 21.
Minute 022: Departure of customer 9 at minute 22.

Minute 025: Arrival of customer 10 at minute 25
Minute 025: Customer 10 has 5 items in hand, and Customer 10 queue at the counter 1
Minute 025: Service for customer 10 started at minute 25.
Minute 030: Departure of customer 10 at minute 30.

Minute 031: Arrival of customer 11 at minute 31
Minute 031: Customer 11 has 4 items in hand, and Customer 11 queue at the counter 4
Minute 031: Service for customer 11 started at minute 31.
Minute 032: Departure of customer 11 at minute 32.

Minute 032: Arrival of customer 12 at minute 32
Minute 032: Customer 12 has 9 items in hand, and Customer 12 queue at the counter 3
Minute 032: Service for customer 12 started at minute 32.

Minute 034: Arrival of customer 13 at minute 34
Minute 034: Customer 13 has 1 items in hand, and Customer 13 queue at the counter 4
Minute 034: Service for customer 13 started at minute 34.

Minute 036: Arrival of customer 14 at minute 36
Minute 036: Customer 14 has 8 items in hand, and Customer 14 queue at the counter 2
Minute 036: Service for customer 14 started at minute 36.
Minute 037: Departure of customer 13 at minute 37.
Minute 038: Departure of customer 12 at minute 38.
Minute 040: Departure of customer 14 at minute 40.

Minute 041: Arrival of customer 15 at minute 41
Minute 041: Customer 15 has 5 items in hand, and Customer 15 queue at the counter 1
Minute 041: Service for customer 15 started at minute 41.

Minute 042: Arrival of customer 16 at minute 42
Minute 042: Customer 16 has 4 items in hand, and Customer 16 queue at the counter 3
Minute 042: Service for customer 16 started at minute 42.
```

```
Minute 044: Arrival of customer 17 at minute 44
Minute 044: Customer 17 has 9 items in hand, and Customer 17 queue at the counter 2
Minute 044: Service for customer 17 started at minute 44.
Minute 045: Departure of customer 15 at minute 45.
Minute 047: Departure of customer 16 at minute 47.

Minute 048: Arrival of customer 18 at minute 48
Minute 048: Customer 18 has 1 items in hand, and Customer 18 queue at the counter 4
Minute 048: Service for customer 18 started at minute 48.
Minute 049: Departure of customer 18 at minute 49.
Minute 050: Departure of customer 17 at minute 50.

Minute 050: Arrival of customer 19 at minute 50
Minute 050: Customer 19 has 8 items in hand, and Customer 19 queue at the counter 1
Minute 050: Service for customer 19 started at minute 50.

Minute 052: Arrival of customer 20 at minute 52
Minute 052: Customer 20 has 5 items in hand, and Customer 20 queue at the counter 3
Minute 052: Service for customer 20 started at minute 52.
Minute 053: Departure of customer 19 at minute 53.
Minute 055: Departure of customer 20 at minute 55.
```

Figure 4.4.1 Real-time Display of Customer Information with Time Stamp

The simulation unfolds in a step-by-step manner, progressing minute by minute, providing a dynamic representation of the system's activities. Throughout the simulation, you witness various pivotal events taking place, including the arrival of customers, the initiation of service for each customer, and the subsequent departure of customers from the counters. This detailed simulation output allows you to closely observe and analyze the queueing behaviour at each counter, gaining valuable insights into the dynamics of customer flow and the overall efficiency of the system.

4.5 Customer Counter Table

Counter 1 (Normal)

Cust No	RN Service time	Service time	Time Service Begin	Time Service Ends	Waiting Time	Time spent in System
2	68	5	2	7	0	5
7	59	5	13	18	0	5
10	71	5	25	30	0	5
15	28	4	41	45	0	4
19	25	3	50	53	0	3

Counter 2 (Normal)

Cust No	RN Service time	Service time	Time Service Begin	Time Service Ends	Waiting Time	Time spent in System
3	70	4	4	8	0	4
5	31	3	8	11	1	4
9	13	2	20	22	0	2
14	55	4	36	40	0	4
17	100	6	44	50	0	6

Counter 3 (Normal)

Cust No	RN Service time	Service time	Time Service Begin	Time Service Ends	Waiting Time	Time spent in System
4	92	7	5	12	0	7
8	72	6	15	21	0	6
12	51	6	32	38	0	6
16	34	5	42	47	0	5
20	1	3	52	55	0	3

Counter 4 (Express)

Cust No	RN Service time	Service time	Time Service Begin	Time Service Ends	Waiting Time	Time spent in System
1	77	3	0	3	0	3
6	67	3	11	14	0	3
11	2	1	31	32	0	1
13	85	3	34	37	0	3
18	18	1	48	49	0	1

Figure 4.5.1 Customer Counter Table Showing Assignment to Respective Counters (Express/Normal)

After the simulation concludes, the output presents comprehensive statistics for each counter in the queuing system. In total, there are three normal counters designated for customers with lower priority, while an express counter is exclusively available for customers with fewer than 5 items. For each counter, the output showcases a table containing detailed information about the customers served at that specific counter. This includes the customer number, the randomly generated number representing the service time, the actual duration of the service, the time at which the service begins, the time at which the service ends, the waiting time experienced by the customer, and the total time spent within the system. These detailed statistics offer valuable insights into the performance and efficiency of each counter in handling customer traffic.

4.6 Result Evaluations

```
The average waiting time:  0.0500

The average interarrival time spent:  2.7368

The average time spent in system:  4.0000

The average service time for counter 1:  4.4000

The average service time for counter 2:  3.8000

The average service time for counter 3:  5.4000

The average service time for counter 4:  2.2000

The probability of a customer having to wait:  0.0500

--> 
```

Figure 4.6.1 Result Evaluation Outputs

By analyzing the output, you can gain valuable insights into the queueing behaviour exhibited in the system. The analysis allows you to observe key metrics such as the average waiting times, average interarrival time, average time spent in the system, average service time for each counter, and the probability of customers having to wait in the queue. These metrics provide a comprehensive understanding of the efficiency and effectiveness of the queuing system, allowing you to assess the overall performance and make informed decisions for process optimization and customer satisfaction improvements.