IE 306 / HOMEWORK 2 / FATMANUR YAMAN / 2019402204 / 01.11.2022

1. Q1)

t_event	Arrival	Departure	Server1	Server2	Interarrival Time	Service Time 1	Service Time 2	Arrival	Departure 1	Departure 2	Time In Queue
0	0,380	0,496	1	0	9,8	17,472	-	9,8	17,472	inf	0
9,8	0,832	0,391	1	1	14,32	-	12	24,12	17,472	21,8	0
17,472			0	1	-	-	-	24,12	inf	21,8	0
21,8	0,020	0,480	0	0	6,2	-	-	24,12	inf	inf	0
24,12	0,975	0,759	1	0	15,75	17,36	-	30,32	41,48	inf	0
30,32			1	1	-	-	22	46,07	41,48	52,32	0
35			1	1	-	-	-	46,07	41,48	52,32	0

2. Q2.a)

In [1]:
```python
import numpy as np
class Simulation:
    def __init__(self, random_number_list,time):

        self.i = 0 #i for counting the index of the random_number_list
        self.random_number_list = random_number_list #random list given in the question
        self.time = time #the time limit

        self.num_in_first = 1 #number of the people in Server 1
        self.num_in_second = 0 #number of the people in Server 2
        self.num_in_q = 0 #number of the people in the queue

        self.clock = 0
        self.t_arrival = self.generate_interarrival() #generate the first arrival
        self.t_departure_first = self.generate_service_first() #generate the first departure in Server 1
        self.t_departure_second = float('inf') #Server 2 is idle.

        self.t_service_first = 0 #service time of Service 1
        self.t_service_second = 0 #service time of Service 2

        self.num_arrivals = 0 #number of arrivals
        self.num_departs = 0 #number of departures
        self.total_wait = 0 #total waiting time of the people
        self.total_spent = 0 #total time spent in the system of people
        self.no_wait = 0
        self.avg_num_cust = 0 #average number of people im the system
        self.avg_wait = 0 #average waiting time
        self.first_server_usage = 0 #time of the service given by Service 1
        self.second_server_usage = 0 #time of the service given by Service 2
        self.first_server_util = 0 #utilization of Server 1
        self.second_server_util = 0 #utilization of Server 2
```

```python
    def simul(self):
        while self.clock <= self.time:
            t_event = min(self.t_arrival, self.t_departure_first, self.t_departure_second)
            #the time for the decision(departure or arrival)
            if self.num_in_q == 0:
                self.no_wait += (t_event - self.clock)

            self.total_spent += (self.num_in_q + self.num_in_first + self.num_in_second)*(t_event - self.clock) #total time spent
            self.first_server_usage += (t_event - self.clock)*self.num_in_first #updates the usage time of Server 1
            self.second_server_usage += (t_event - self.clock)*self.num_in_second #updates the usage time of Server 2
            self.total_wait += self.num_in_q*(t_event - self.clock) #updates the total waiting times
            if self.time == 35:
                print('Arrival: ' + str(self.t_arrival) + '\n' +
                      'Departure 1: ' + str(self.t_departure_first) + '\n' +
                      'Departure 2: ' + str(self.t_departure_second) + '\n')

            self.clock = t_event #jumps when the new event occurs

            self.first_server_util = self.first_server_usage / self.clock #updates the utilization of Server 1
            self.second_server_util = self.second_server_usage / self.clock #updates the utilization of Server 2
            self.avg_wait = self.total_wait / self.clock #calculates average waiting time
            self.avg_num_cust = self.total_spent/self.clock #calculates the average  number of people in the system

            #Determining what to do next:
            #If the minimum is arrival, the arrival event occurs.
            if t_event == self.t_arrival:
                self.handle_arrival_event()
            #If the minimum is departure, the departure will occur and the server will be idle(inf).
            if t_event == self.t_departure_second:
                self.t_departure_second = float('inf')
            if t_event == self.t_departure_first:
                self.t_departure_first = float('inf')
```

```
        #the number of the people in each server
        #This indicates whether the server is idle or not.
        if self.t_departure_first == float('inf'):
            self.num_in_first = 0
        else:
            self.num_in_first = 1

        if self.t_departure_second == float('inf'):
            self.num_in_second = 0
        else:
            self.num_in_second = 1

    return self.write_results() #Write the results when the time is being exceeded.


def handle_arrival_event(self):
    self.num_arrivals += 1 #increase the number of arrivals

    self.t_arrival = self.clock + self.generate_interarrival() #generate the next interarrival and arrival time

    #The if loop below determines whether the server is idle or not, then chooses the approprìote one.
    if self.t_departure_first == float('inf') and self.t_departure_second == float('inf'):
        #If both servers are idle, choose Server 1.
        return self.handle_departure_event_first()
    if self.t_departure_first != float('inf') and self.t_departure_second == float('inf'):
        return self.handle_departure_event_second()
    if self.t_departure_first == float('inf') and self.t_departure_second != float('inf'):
        return self.handle_departure_event_first()
    if self.t_departure_first != float('inf') and self.t_departure_second != float('inf'):
        #If both servers are busy, increment the numbers of the people in the queue by one.
        self.num_in_q += 1
```

```
def handle_departure_event_first(self):
    self.num_departs += 1 #increase the number of departures
    self.t_departure_first = self.clock + self.generate_service_first() #generate the departure time
    if self.num_in_q >= 1:
        self.num_in_q -= 1 #decrease the number of the people in the queue since it is departure
    return self.t_departure_first

def handle_departure_event_second(self):
    self.num_departs += 1
    self.t_departure_second = self.clock + self.generate_service_second() #generate the departure time
    if self.num_in_q >= 1:
        self.num_in_q -= 1 #decrease the number of the people in the queue since it is departure
    return self.t_departure_second

def generate_interarrival(self):
    self.t_interarrival =  self.random_number_list[self.i]*10+6 #generates the interarrival times by U(6,16)
    self.i += 1
    return self.t_interarrival


def generate_service_first(self):
    self.t_service_first = self.random_number_list[self.i]*7+14 #generates the service times for Server 1 by U(14,21)
    self.i += 1
    return self.t_service_first
```

```
def generate_service_second(self): #generates the service times for Server 2 by using the CDF's of the given distirbution
    if self.random_number_list[self.i] <= 0.18:
        self.t_service_second =  8
        self.i += 1
    elif 0.18 < self.random_number_list[self.i] <= 0.48:
        self.t_service_second = 12
        self.i += 1
    elif 0.48 < self.random_number_list[self.i] <= 0.78:
        self.t_service_second = 22
        self.i += 1
    elif 0.78 < self.random_number_list[self.i] <=1:
        self.t_service_second = 33
        self.i += 1

    return self.t_service_second

def write_results(self):
    if self.time <= self.clock:
        print('Average Time Spent In The Queue: ' + str(self.avg_wait) + '\n'+
        'The Average Number of Customer In The System: ' + str(self.avg_num_cust) +'\n'+
        'The Average Utilization of Server 1: ' + str(self.first_server_util) +'\n'+
        'The Average Utilization of Server 2: ' + str(self.second_server_util)+ '\n' +
        'The Possibility of a Customer Not Waiting In The Queue: '+ str(self.no_wait/self.clock) + '\n')
```

In [2]:
```
random_number_list = [0.38, 0.496, 0.832, 0.391, 0.020, 0.480, 0.975, 0.759, 0.905, 0.593, 0.560]
s = Simulation(random_number_list,35)
s.simul()
```

```
Arrival: 9.8
Departure 1: 17.472
Departure 2: inf

Arrival: 24.12
Departure 1: 17.472
Departure 2: 21.8

Arrival: 24.12
Departure 1: inf
Departure 2: 21.8

Arrival: 24.12
Departure 1: inf
Departure 2: inf

Arrival: 30.32
Departure 1: 41.480000000000004
Departure 2: inf

Arrival: 46.07
Departure 1: 41.480000000000004
Departure 2: 52.32

Average Time Spent In The Queue: 0.0
The Average Number of Customer In The System: 1.3980713596914176
The Average Utilization of Server 1: 0.8397299903567985
The Average Utilization of Server 2: 0.5583413693346191
The Possibility of a Customer Not Waiting In The Queue: 1.0
```

3. Q2.b)

In [3]:
```
seed_1 = list(np.random.randn(10000))
s1 = Simulation(seed_1,7000)
s1.simul()
```

```
Average Time Spent In The Queue: 10.984344468479993
The Average Number of Customer In The System: 12.076708687456954
The Average Utilization of Server 1: 0.6317668076110214
The Average Utilization of Server 2: 0.46059741136593496
The Possibility of a Customer Not Waiting In The Queue: 0.16242291377615908
```

In [4]:
```
seed_2 = list(np.random.randn(10000))
s2 = Simulation(seed_2,7000)
s2.simul()
```

```
Average Time Spent In The Queue: 22.324195690214758
The Average Number of Customer In The System: 23.39509280275036
The Average Utilization of Server 1: 0.6344001212231585
The Average Utilization of Server 2: 0.43649699131245906
The Possibility of a Customer Not Waiting In The Queue: 0.10053294926669847
```

In [5]:
```
seed_3 = list(np.random.randn(10000))
s3 = Simulation(seed_3,7000)
s3.simul()
```

```
Average Time Spent In The Queue: 9.706868128457241
The Average Number of Customer In The System: 10.826138650496729
The Average Utilization of Server 1: 0.6354980924466792
The Average Utilization of Server 2: 0.4837724295928371
The Possibility of a Customer Not Waiting In The Queue: 0.11175033640814977
```

In [6]:
```
seed_4 = list(np.random.randn(10000))
s4 = Simulation(seed_4,7000)
s4.simul()
```

```
Average Time Spent In The Queue: 8.60151767054128
The Average Number of Customer In The System: 9.703734721925208
The Average Utilization of Server 1: 0.6490421078258277
The Average Utilization of Server 2: 0.4531749435580918
The Possibility of a Customer Not Waiting In The Queue: 0.2364244580616939
```

Little's Law

$$L = \lambda \times W$$

$$Average\ number\ of\ customers = Average\ Number\ of\ Arrivals \times Average\ Time\ Spent\ In\ The\ System$$

```
In [7]: s4.num_arrivals
Out[7]: 995

In [8]: s4.total_spent
Out[8]: 67964.15919549279

In [9]: s4.avg_num_cust
Out[9]: 9.703734721925208

In [10]: (s4.num_arrivals/s4.clock) * (s4.total_spent/s4.num_arrivals)
Out[10]: 9.703734721925207
```

As can be seen above, Little's Law holds for my simulations.