BIOSTAT213 Final Project

Abdullah-Al-Zubaer Imran

Part1

Given the previous assumptions, from the previous results it was found that having one line for 3 tellers had a slightly smaller expected waiting time than the queuing system where each teller had their own line. Is there a way to make these 2 queuing systems equivalent? That is can we design a mechanism such that the three-line system is equivalent to the one-line system. Fir example, what if we let people switch lines but only allow the person who had been waiting the longest to switch? What if we let the person who just arrived switch?

→ In order to design a mechanism so that the one-line queuing system becomes equivalent to the three-line queuing system, we tried three possible approaches. The first approach is imposing first-come-first-served policy in the three-line queuing system. The second approach is providing service in the servers based on the order of estimated service time (possibly by issuing tokens at the time of arrival for each of the customers). And the third approach is by allowing switching of customer from one line to another. [Complete source code has been put at the end of this report]

Table 1: Illustration of results for approach1 (waiting longer being switched)

Replications	Estimated waiting time	
	Single-line system	Three-line system
1	0.1900326	0.1936199
2	0.1900955	0.1935523
3	0.1907698	0.1943637
4	0.1896634	0.1939655
5	0.1904731	0.1928338
6	0.1898786	0.1938059
7	0.1900217	0.1929638
8	0.1904560	0.1929870
9	0.1903990	0.1937690
10	0.1902204	0.1942945
11	0.1903864	0.1938671
12	0.1902545	0.1930802
13	0.1900762	0.1927826
14	0.1905167	0.1942708
15	0.1898705	0.1934112
16	0.1909083	0.1929046
17	0.1898817	0.1938820
18	0.1904910	0.1929896
19	0.1900913	0.1932806
20	0.1911238	0.1930813
21	0.1903388	0.1940354
22	0.1903233	0.1925694
23	0.1900285	0.1934034
24	0.1902824	0.1938784
25	0.1904527	0.1934195

For the first approach, we added a subroutine for determining the customer from the waiting line(s) who has been waiting for the longest. Excluding the customers already in the system taking service from the three tellers and the customers already departed, we could track the customer in the waiting line(s) who has been waiting for the longest. We run the simulation with 25 replications and 100000 repetitions for each replica. Although for the single-line queuing system, the estimated waiting time still lower than the three-line system, with same combination of customer arrivals and relatively greater number of repetitions, the estimated waiting time could almost be same for both queuing system.

Table 2: Illustration of results for approach2 (token based on estimated service completion time)

Replications	**Estimated waiting time**	
	Single-line system	**Three-line system**
1	0.1902189	0.1955093
2	0.1897155	0.1931007
3	0.1908261	0.1938535
4	0.1884816	0.1953644
5	0.1924608	0.1950341
6	0.1916709	0.1934156
7	0.1910275	0.1931910
8	0.1901258	0.1938609
9	0.1880510	0.1936641
10	0.1880487	0.1938903
11	0.1905127	0.1938483
12	0.1910106	0.1945841
13	0.1891092	0.1925056
14	0.1903553	0.1898522
15	0.1901236	0.1916816
16	0.1895184	0.1907043
17	0.1922176	0.1934970
18	0.1882334	0.1935894
19	0.1900648	0.1934949
20	0.1888692	0.1954434
21	0.1889591	0.1937990
22	0.1904510	0.1909130
23	0.1909657	0.1945654
24	0.1915724	0.1940485
25	0.1899674	0.1922243

For the second approach (appraoch2) as has been reported on Table 2, we assigned tokens by means of estimated service completion time for each of the customers at the times of their arrivals. With this, we could design an efficient multi-line queuing system which would perform similar to the single-line system. As we see from the tabulated results, the expected waiting time for the three-line queuing system became pretty close to the single-line system. We run the simulation for 1000 repetitions and generated 25 replicas of the simulation considering the time in running the program. With more repetitions, we could see a real match or even better performance in the three-line queuing system.

Table 3: Illustration of results for approach3 (allowing customers to switch between lines)

Replications	Estimated waiting time	
	Single-line system	Three-line system
1	0.1906348	0.1948623
2	0.1901456	0.1944325
3	0.1908379	0.1954282
4	0.1910901	0.1922760
5	0.1910734	0.1958650
6	0.1915559	0.1949458
7	0.1891968	0.1941614
8	0.1919637	0.1927295
9	0.1898462	0.1957100
10	0.1894479	0.1929438
11	0.1905310	0.1945480
12	0.1899735	0.1928569
13	0.1904331	0.1926474
14	0.1904395	0.1925491
15	0.1900046	0.1942300
16	0.1899115	0.1950331
17	0.1906591	0.1934587
18	0.1920066	0.1922881
19	0.1888651	0.1935525
20	0.1903624	0.1947932
21	0.1914623	0.1942933
22	0.1899853	0.1925061
23	0.1877700	0.1935836
24	0.1899837	0.1943584
25	0.1897313	0.1929649

It is quite obvious from the approach 3 (as has been illustrated in Table 3), allowing the switching for the customer just arrived even makes the things worse. Rather than the estimated waiting time being improved, it becomes greater than the single-line queuing system for the three-line system. As the customers coming earlier have to wait longer, the estimated waiting time for the three-line system cannot be matched to the single-line queuing system.

Part2

In the tandem system, does it matter where you put the slowest teller? That is, does the expected waiting time change if you the slowest teller is teller 1 or teller 2 or teller 3?

→In order figure it out, the tandem system with all the previous assumptions was run with 50000 repetitions. We generated 25 replicas for each occasion when the slowest teller is Teller1 or Teller2 or Teller3. Based on our result, it could be concluded that the position of the slowest teller in the sequence of tellers in a tandem system doesn't have any impact on the change of the expected waiting time. Since, each customer needs to take service from the slowest teller anyway, it doesn't matter. Following table based on the result supports the claim. If we look at the tabulated results, more or less the expected waiting time in each case is approx. 2.27. There is slight chance that if Teller 1 is the slowest, then the waiting time could be greater.

Table 4: Illustration of results assuming different positions of the slowest teller in the system

Replications	**Expected Waiting Time when the slowest teller is:**		
	Teller 1	**Teller 2**	**Teller 3**
1	2.272803	2.279176	2.267609
2	2.272341	2.271175	2.275577
3	2.269915	2.264362	2.272503
4	2.273145	2.271503	2.275332
5	2.269171	2.274902	2.265873
6	2.271732	2.278283	2.267342
7	2.269828	2.264103	2.273797
8	2.274877	2.273097	2.269821
9	2.272997	2.274476	2.270885
10	2.278580	2.268203	2.278516
11	2.276716	2.269990	2.272169
12	2.268264	2.266887	2.275030
13	2.278511	2.276190	2.273256
14	2.278224	2.271678	2.268987
15	2.276317	2.267423	2.269575
16	2.271730	2.272579	2.261038
17	2.270866	2.269311	2.273056
18	2.273805	2.276535	2.265794
19	2.273292	2.267231	2.263021
20	2.269879	2.269621	2.265141
21	2.269557	2.264368	2.264069
22	2.272711	2.270449	2.267174
23	2.270367	2.270709	2.271663
24	2.271669	2.268469	2.272685
25	2.271689	2.270709	2.268429
Average	**2.2728**	**2.271**	**2.27**

Source Code

Part1

Approach1:

```
## subroutine ##
## intensity rate in which the customers arrive ##
lambda_t <- function(time)      # the banks open from 8am to 5pm, so we set
        {
                if(time >= 4 && time <= 5)     # the time from 0 to 9 and the rate is 4/hr
                        {
                                lambda <- 6                 # but increase to 6 between 4 and 5.
                                }
                else
                 {
                        lambda <- 4
                        }
                list(lamt=lambda)
                }

### subroutine ###
### Generate Tt which will be assigned to ta: next arrival time ###
generate_Tt <- function(ta,t1,t2,t3,t)
        {
                Tt <- t - (1/lambda_t(t)$lamt)*log(runif(1))    # next arrival time

                if (ta < 4)                    # ta is less than 4
                {
                        ta <- Tt
                        if (ta > 4)       #if greater than 4, change rate
```

```
                    {
        ta <- 4+((ta-4)*4/6)
         }
          }


          else if ( 4 <= ta && ta <= 5 )
               {
                    # 4 <= ta <= 5
                    ta <- Tt
                    if (ta > 5)
                     {
                         ta <- 5+((ta-5)*6/4)      #if greater than 5 , change rate
                         }
                    }
          else
               ta <- Tt
          if ( ta > 9)
               ta <- 99999     # To insure that nobody can enter the system after 5 pm
          list(ta=ta)
     }
###Subroutine to determine the next customer to be served in the system from the current waiting line(s)
 ##It takes inputs: customers currently in the system being served, latest departed list of customers,
   #latest arrival list of customers
  ##It returns the next customer based on the waiting time
 next_customer <- function(s1, s2, s3, depart, arriv)
 {
     ins <- union(s1, union(s2, s3))
     ins_depart <- union(ins, which(depart!=FALSE))
```

```
        in_line <- setdiff(which(arriv!=FALSE), ins_depart)
        next_cust <- which(arriv ==min(arriv[in_line]))
        next_cust = next_cust
 }
replica1 <- numeric()
replica3 <- numeric()

### Situation ONE: One line with parallel servers
for (i in 1:25)
{
n.rep <- 10000
ave.1 <- numeric(n.rep)

for (j in 1:n.rep)
 {
        ## Main Program ##
        ## Initialization ##
        t <- 0          # current time
        T <- 9          # after T, no one can enter this system
        n <- 0          # number of people is in the system in the time t
        Na <- 0                 # total number of arrival

        C1 <- C2 <- C3 <- 0     # Ci is the number of customers served by i by time t
        Nd <- 0                 # total number of departure
        T0 <- numeric(1)
        r1 <- 6         # the service time for server i has exponential distn with rate ri
        r2 <- 5
        r3 <- 4
```

```
lambda <- 6
SS <- c(0,0,0,0)          # SS(n,i1,i2,i3) n:number of customers in the system
                                      # ij: person i is being served by server j

t <- t- (1/lambda_t(t)$lamt)*log(runif(1))      # generate the first arrival time

ta <- T0 <- t              # next arrival
t1 <- t2 <- t3 <- 99999            # ti is the service completion time of the customer,
              #presently being served by server i

ANa <- numeric()       # the arrival time of customer n
DNd <- numeric()       # the departure time of customer n

repeat #repeat starts
{
        if(ta==99999 && ta==t1 && t1==t2 && t2==t3)   #if0 starts
                        break         #if0 ends  #Abnormal termination

        ### CASE 1

        if (ta == min(ta,t1,t2,t3))        #if1 starts within repeat  #Case 1
         {
                t <- ta
                Na <- Na + 1
    ANa[Na] <- t
```

```
ta <- generate_Tt(ta,t1,t2,t3,t)$ta        # Notice: the change of rate

if (SS[1] == 0)  #if2 starts within if1              #nobody is in the system
 {
        Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
        #print(Ns)
        #cat("\n")
        SS <- c(1,Ns,0,0)                    # go to server 1 if all are free
        t1 <- t - (1/r1)*log(runif(1))     # the time of service being completed
        }   #if2 ends

else if (SS[1] == 1)     #elseif1 starts within if1 #only one is in the system
 {
         SS[1] <- 2
         if(SS[3]==0 && SS[4]==0) #if2 starts within elseif1
                {

                Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
        #print(Ns)
            #cat("\n")
                SS[3] <- Ns
                t2 <- t-(1/r2)*log(runif(1))
                }   #if2 ends

 else if (SS[2] == 0 && SS[4] == 0) #elseif2 of if2 within elseif1
         {
```

```
                    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
                        #print(Ns)
                      #cat("\n")
                        SS[2] <- Ns
                        t1 <- t-(1/r1)*log(runif(1))
                        }     #elseif2 ends

                     else if (SS[2] == 0 && SS[3] == 0) #elseif3 of if2 within elseif1
                      {
                    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
                        #print(Ns)
                      #cat("\n")
                        SS[2] <- Ns
                        t1 <- t-(1/r1)*log(runif(1))
                        }      #elseif3 ends
                    }   #elseif1 ends

          else if (SS[1] == 2) #elseif4 starts within if1 #two customers are in the system
               {
                     SS[1] <- 3
                    if(SS[4]==0) #if3 starts within elseif4
                        {         Ns  <-  next_customer(SS[2],  SS[3],  SS[4],  DNd,
ANa)                 #next customer who has been waiting longest
                             #print(Ns)
                        #cat("\n")
                             SS[4] <- Ns
                             t3 <- t-(1/r3)*log(runif(1))
```

```
                    }    #if3 ends

            else if (SS[3] == 0)   #elseif5 starts within elseif4
                {
             Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
                 #print(Ns)
               #cat("\n")
                 SS[3] <- Ns
                 t2 <- t-(1/r2)*log(runif(1))
                 }      #elseif5 ends

            else if (SS[2] == 0)    #elseif6 starts within elseif4
                {
             Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
#next customer who has been waiting longest
                 #print(Ns)
               #cat("\n")
                 SS[2] <- Ns
                 t1 <- t-(1/r1)*log(runif(1))
                 }     #elseif6 ends

         }       #elseif4 ends

        else if (SS[1] > 2)       #elseif7 starts within if1 #more than 2 customers
            {
                SS[1] <- SS[1]+1
                }    #elseif7 ends
```

```
            }    #if1 ends


### CASE 2
        else if ( t1 == min(ta,t1,t2,t3))  #elseif8 starts  #Case 2
        {
            t <- t1
            C1 <- C1 + 1                          # number of customers served by 1
            DNd[SS[2]] <- t                       # the departure time for SS[2]



            if (SS[1]==1)  #if4 starts  within elseif8                  # one customer in SS
              {
                t1 <- 99999

                SS <- c(0,0,0,0)
          }        #if4 ends

            else if (SS[1]==2) #elseif9 starts within elseif8                #          Two
customer
            {
                t1 <- 99999
                if (SS[4] == 0) #if5 starts within elseif9
                 {
                    SS <- c(1,0,SS[3],0)
                    }      #if5 ends

                else if ( SS[3] == 0)   #elseif10 starts within elseif9
                   {
                     SS <- c(1,0,0,SS[4])
```

```
                } #elseif10 ends

               } #elseif9 ends

        else if (SS[1]==3) #elseif11 starts within elseif8          # Three Customer
        {
            t1 <-99999
            SS <- c(2,0,SS[3],SS[4])
            }     #elseif11 ends

        else if(SS[1] >3) #elseif12 starts within elseif8          # more than three customer
         {
            #m <- max(SS[2],SS[3],SS[4])
            Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
            #print(Ns)
            #cat("\n")
            SS <- c((SS[1]-1), Ns, SS[3], SS[4])
            t1 <- t-1/r1*log(runif(1))
            }  #elseif12 ends

     }   #elseif8 ends
    ### CASE 3
    else if (t2 == min(ta,t1,t2,t3)) #elseif13 starts # Case 3
        {
            t <- t2
            C2 <- C2 + 1                    # number of customers served by 2
            DNd[SS[3]] <- t                 # the departure time for SS[3]
```

```
if (SS[1]==1) #if6 starts within elseif13                    # one customer in SS
{
        t2 <- 99999
        SS <- c(0,0,0,0)
}     #if6 ends

else if (SS[1]==2) #elseif14 of if6 starts within elseif13
# Two customer
 {
        t2 <- 99999
        if (SS[4] == 0) #if7 starts within elseif14
         {
                SS <- c(1,SS[2],0,0)
                } #if7 ends
        else if ( SS[2] == 0)    #elseif15 of if7 starts within elseif14
          {
                SS <- c(1,0,0,SS[4])
   }   #elseif15 ends

         }     #elseif14 ends

else if (SS[1]==3)  #elseif16 starts within elseif13      # Three Customer
 {
        t2 <-99999
        SS <- c(2,SS[2],0,SS[4])
        }     #elseif16 ends
```

```
                  else if(SS[1] >3)   #elseif17 starts within elseif13                        # more
than three customer
                   {
                        #m <- max(SS[2],SS[3],SS[4])
                        Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
                        print(Ns)
                        cat("\n")
                        SS <- c((SS[1]-1), SS[2], Ns, SS[4])
                        t2 <- t-1/r2*log(runif(1))
                        }   #elseif17 ends
          } #elseif13 ends

      ### CASE 4
      else if ( t3 == min(ta,t1,t2,t3))   #elseif18 starts    # Case 4
      {
             t <- t3
            C3 <- C3 + 1                              # number of customers served by 3
            DNd[SS[4]] <- t                     # the departure time for SS[4]

             if (SS[1]==1)   #if8 starts within elseif18       # one customer in SS
              {
                  t3 <- 99999
                  SS <- c(0,0,0,0)
                  }    #if8 ends
            else if (SS[1]==2) #elseif19 starts within elseif18                        #          Two
customer
              {
                  t3 <- 99999
                  if (SS[3] == 0) #if9 starts within elseif19
```

```
                    {
                              SS <- c(1,SS[2],0,0)
                               }    #if9 ends
         else if ( SS[2] == 0)     #elseif20 starts within elseif19
                              {
                               SS <- c(1,0,SS[3],0)
                                 }      #elseif20 ends
                    }    #elseif18 ends
          else if (SS[1]==3)    #elseif21 starts within elseif18                    #           Three
Customer
     {
                    t3 <-99999
                    SS <- c(2,SS[2],SS[3],0)
                    }     #elseif21 ends
          else if(SS[1] >3)   #elseif22 starts within elseif18                    #   more   than
three customer
          {
                    #m <- max(SS[2],SS[3],SS[4])
                    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
                    print(Ns)
                    cat("\n")
                    SS <- c((SS[1]-1), SS[2], SS[3], Ns)
                    t3 <- t-1/r3*log(runif(1))
                    }       #elseif22 ends
     }      #elseif18 ends
  } #repeat1 ends
     ave.1[j]<- mean(DNd-ANa)
 }     #for-loop1 ends
 mean(ave.1)
```

```
replica1[i] <- mean(ave.1)



### Situation THREE: each teller has their own line
n.rep <- 10000
ave.3 <- numeric(n.rep)
for (j in 1:n.rep)
 {
 ### Main Program ###
 ## Initialization ##
 t <- 0              # current time
 T <- 9              # after T, no one can enter this system
 Na <- 0                    # total number of arrival
 S1 <- S2 <- S3 <- numeric()   # Si is the sequence of customers in the server i at time t
 Nd <- 0                    # total number of departure
 T0 <- numeric(1)
 r1 <- 6             # the service time for server i has exponential distn with rate ri
 r2 <- 5
 r3 <- 4
 t <- t- (1/lambda_t(t)$lamt)*log(runif(1))      # generate the first arrival time
 ta <- T0 <- t              # next arrival
 t1 <- t2 <- t3 <- 99999           # ti is the service completion time of the customer
                                                      # presently being served by server i
 ANa <- numeric()      # the arrival time of customer n
 DNd <- numeric()      # the departure time of customer n
 repeat {
         if(ta==99999 && ta==t1 && t1==t2 && t2==t3)
                        break
```

```
        ### CASE 1
if (ta == min(ta,t1,t2,t3))        # Case 1
  {
        t <- ta
        Na <- Na + 1
        ta <- generate_Tt(ta,t1,t2,t3,t)$ta  # new arrival time
        ANa[Na] <- t
if (length(S1) == min(length(S1),length(S2),length(S3)))     # enter 1
   {
        Ns <- next_customer(S1, S2, S3, DNd, ANa)

        S1[length(S1)+1] <- Ns         # customer Ns enter the server 1

        if (length(S1) == 1)
          {
                t1 <- t - (1/r1)*log(runif(1))    # the time of server being completed
                }
        }
else if (length(S2) == min(length(S1),length(S2),length(S3)))          # enter 2
        {
         Ns <- next_customer(S1, S2, S3, DNd, ANa)
         S2[length(S2)+1] <- Ns        # customer Ns enter the sever 2
         if (length(S2) == 1)
          {
                t2 <- t - (1/r2)*log(runif(1))    # the time of server being completed
                  }
          }
```

```
    else if (length(S3) == min(length(S1),length(S2),length(S3)))          # enter 2
            {
             Ns <- next_customer(S1, S2, S3, DNd, ANa)
             S3[length(S3)+1] <- Ns         # customer Ns enter the sever 3
             if (length(S3) == 1)
                     {
                     t3 <- t - (1/r3)*log(runif(1))   # the time of server being completed
                      }
         }
  }
```

```
### CASE 2
    else if ( t1 == min(ta,t1,t2,t3))  # Case 2
  {
            t <- t1
            DNd[S1[1]] <- t1        # The first in server will leave
            S1 <- S1[-1]     # the first leave
            if (length(S1) == 0)
                    t1 <- 99999
            if (length(S1) > 0)
                    t1 <- t - (1/r1)*log(runif(1))     # the time of server being completed
      }
```

```
### CASE 3
    else if ( t2 == min(ta,t1,t2,t3))  # Case 3
     {
            t <- t2
            DNd[S2[1]] <- t2        # The first in server will leave
```

```
            S2 <- S2[-1]     # the first leave
            if (length(S2) == 0)
                  t2 <- 99999

            if (length(S2) > 0)
                  t2 <- t - (1/r2)*log(runif(1))     # the time of server being completed
      }

### CASE 4
 else if ( t3 == min(ta,t1,t2,t3))  # Case 4
  {
            t <- t3
            DNd[S3[1]] <- t3          # The first in server will leave
            S3 <- S3[-1]     # the first leave
            if (length(S3) == 0)
                  t3 <- 99999
            if (length(S3) > 0)
                  t3 <- t - (1/r3)*log(runif(1))     # the time of server being completed
            }
 }     #repeat ends
ave.3[j] <- mean(DNd-ANa)
 }   #for-loop ends
mean(ave.3)
replica3[i] <- mean(ave.3)
}
```

Approach2:

```
## subroutine ##
## intensity rate in which the customers arrive ##
lambda_t <- function(time)     # the banks open from 8am to 5pm, so we set
        {
                if(time >= 4 && time <= 5)    # the time from 0 to 9 and the rate is 4/hr
                        {
                                lambda <- 6             # but increase to 6 between 4 and 5.
                                }
                else
                 {
                        lambda <- 4
                        }
                list(lamt=lambda)
                }

### subroutine ###
### Generate Tt which will be assigned to ta: next arrival time ###
generate_Tt <- function(ta,t1,t2,t3,t)
        {
                Tt <- t - (1/lambda_t(t)$lamt)*log(runif(1))    # next arrival time

                if (ta < 4)                 # ta is less than 4
                {
                        ta <- Tt
                        if (ta > 4)      #if greater than 4, change rate
                                {
              ta <- 4+((ta-4)*4/6)
```

```
        }
          }

          else if ( 4 <= ta && ta <= 5 )
                  {
                          # 4 <= ta <= 5
                          ta <- Tt
                          if (ta > 5)
                           {
                                  ta <- 5+((ta-5)*6/4)        #if greater than 5 , change rate
                                  }
                          }
          else
                  ta <- Tt

          if ( ta > 9)
                  ta <- 99999      # To insure that nobody can enter the system after 5 pm

          list(ta=ta)
      }
```

###Subroutine to determine the next customer to be served in the system from the current waiting line(s)

```
 ##It takes inputs: customers currently in the system being served, latest departed list of customers,
   #latest arrival list of customers
  ##It returns the next customer based on the waiting time
 next_customer <- function(s1, s2, s3, depart, arriv)
 {
        ins <- union(s1, union(s2, s3))
```

```
        ins_depart <- union(ins, which(depart!=FALSE))
        in_line <- setdiff(which(arriv!=FALSE), ins_depart)
        next_cust <- which(arriv == min(arriv[in_line]))

        next_cust = next_cust
 }

replica1 <- numeric()
replica3 <- numeric()

for (i in 1:25)
{

### Situation ONE: One line with parallel servers
n.rep <- 1000
ave.1 <- numeric(n.rep)

for (j in 1:n.rep)
 {
        ## Main Program ##
        ## Initialization ##
        t <- 0          # current time
        T <- 9          # after T, no one can enter this system
        n <- 0          # number of people is in the system in the time t
        Na <- 0                 # total number of arrival

        C1 <- C2 <- C3 <- 0     # Ci is the number of customers served by i by time t
        Nd <- 0                 # total number of departure
```

```
T0 <- numeric(1)
r1 <- 6           # the service time for server i has exponential distn with rate ri
r2 <- 5
r3 <- 4

lambda <- 6
SS <- c(0,0,0,0)        # SS(n,i1,i2,i3) n:number of customers in the system
                        # ij: person i is being served by server j

t <- t- (1/lambda_t(t)$lamt)*log(runif(1))     # generate the first arrival time

ta <- T0 <- t           # next arrival
t1 <- t2 <- t3 <- 99999        # ti is the service completion time of the customer,
        #presently being served by server i

ANa <- numeric()        # the arrival time of customer n
DNd <- numeric()        # the departure time of customer n
Est <- numeric()        #estimated time of service at the time of arrival

repeat #repeat starts
{
    if(ta==99999 && ta==t1 && t1==t2 && t2==t3)   #if0 starts
            break       #if0 ends  #Abnormal termination

    ### CASE 1

    if (ta == min(ta,t1,t2,t3))      #if1 starts within repeat  #Case 1
     {
```

```
t <- ta
Na <- Na + 1
ANa[Na] <- t

meanr <- mean(r1, r2, r3)
Est[Na] <- runif(1)*10

ta <- generate_Tt(ta,t1,t2,t3,t)$ta        # NOtice: the change of rate

if (SS[1] == 0)  #if2 starts within if1            # nobody is in the system
 {
    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
    #print(Ns)
    #cat("\n")
    SS <- c(1,Ns,0,0)              # go to server 1 if all are free
    t1 <- t - (1/r1)*log(runif(1))    # the time of service being completed
    }   #if2 ends

else if (SS[1] == 1)    #elseif1 starts within if1       # only one is in the system
 {
    SS[1] <- 2
    if(SS[3]==0 && SS[4]==0) #if2 starts within elseif1
        {

        Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
    #print(Ns)
```

```
#cat("\n")
SS[3] <- Ns
t2 <- t-(1/r2)*log(runif(1))
}   #if2 ends

else if (SS[2] == 0 && SS[4] == 0) #elseif2 of if2 within elseif1
{
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
#print(Ns)
#cat("\n")
SS[2] <- Ns
t1 <- t-(1/r1)*log(runif(1))
}      #elseif2 ends

else if (SS[2] == 0 && SS[3] == 0) #elseif3 of if2 within elseif1
{
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
#print(Ns)
#cat("\n")
SS[2] <- Ns
t1 <- t-(1/r1)*log(runif(1))
}      #elseif3 ends
}   #elseif1 ends

else if (SS[1] == 2)     #elseif4 starts within if1          #  two  customers  are in the system
{
```

```
SS[1] <- 3
if(SS[4]==0) #if3 starts within elseif4
    {       Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
            #print(Ns)
    #cat("\n")
            SS[4] <- Ns
            t3 <- t-(1/r3)*log(runif(1))
            }  #if3 ends

else if (SS[3] == 0)  #elseif5 starts within elseif4
    {
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
    #print(Ns)
  #cat("\n")
    SS[3] <- Ns
    t2 <- t-(1/r2)*log(runif(1))
    }    #elseif5 ends

else if (SS[2] == 0)   #elseif6 starts within elseif4
    {
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est) #next customer who has been waiting longest
    #print(Ns)
  #cat("\n")
    SS[2] <- Ns
    t1 <- t-(1/r1)*log(runif(1))
    }   #elseif6 ends
```

```
                    }        #elseif4 ends


                    else if (SS[1] > 2)       #elseif7 starts within if1                    # more than 2 customers
                        {
                            SS[1] <- SS[1]+1
                            }     #elseif7 ends

                }    #if1 ends
```

```
### CASE 2
        else if ( t1 == min(ta,t1,t2,t3))  #elseif8 starts  #Case 2
        {
            t <- t1
            C1 <- C1 + 1                                 # number of customers served by 1
            DNd[SS[2]] <- t                              # the departure time for SS[2]


            if (SS[1]==1)  #if4 starts  within elseif8                  # one customer in SS
              {
                t1 <- 99999

                SS <- c(0,0,0,0)
```

```
        }        #if4 ends

            else if (SS[1]==2) #elseif9 starts within elseif8                    #           Two
customer
            {
                t1 <- 99999
                if (SS[4] == 0) #if5 starts within elseif9
                 {
                    SS <- c(1,0,SS[3],0)
                    }      #if5 ends

                else if ( SS[3] == 0)   #elseif10 starts within elseif9
                   {
                     SS <- c(1,0,0,SS[4])
                 }    #elseif10 ends

                 }   #elseif9 ends

            else if (SS[1]==3) #elseif11 starts within elseif8                   #          Three
Customer
            {
                t1 <-99999
                SS <- c(2,0,SS[3],SS[4])
                }         #elseif11 ends

            else if(SS[1] >3) #elseif12 starts within elseif8                    #   more   than
three customer
             {
                #m <- max(SS[2],SS[3],SS[4])
```

```
                Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est)
                #print(Ns)
                #cat("\n")
                SS <- c((SS[1]-1), Ns, SS[3], SS[4])
                t1 <- t-1/r1*log(runif(1))
                }   #elseif12 ends

        }     #elseif8 ends





      ### CASE 3
      else if (t2 == min(ta,t1,t2,t3)) #elseif13 starts # Case 3
            {
                t <- t2
                C2 <- C2 + 1                   # number of customers served by 2
                DNd[SS[3]] <- t                # the departure time for SS[3]

                if (SS[1]==1) #if6 starts within elseif13          # one customer in SS
                {
                      t2 <- 99999
                      SS <- c(0,0,0,0)
                }     #if6 ends

                else if (SS[1]==2) #elseif14 of if6 starts within elseif13
# Two customer
                  {
```

```
t2 <- 99999
if (SS[4] == 0) #if7 starts within elseif14
 {
      SS <- c(1,SS[2],0,0)
      } #if7 ends

else if ( SS[2] == 0)    #elseif15 of if7 starts within elseif14
  {
      SS <- c(1,0,0,SS[4])
}   #elseif15 ends

 }     #elseif14 ends

else if (SS[1]==3)  #elseif16 starts within elseif13      # Three Customer
 {
      t2 <-99999
      SS <- c(2,SS[2],0,SS[4])
      }     #elseif16 ends

else if(SS[1] >3)   #elseif17 starts within elseif13                # more
than three customer
 {
      #m <- max(SS[2],SS[3],SS[4])
      Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est)
      print(Ns)
      cat("\n")
      SS <- c((SS[1]-1), SS[2], Ns, SS[4])
      t2 <- t-1/r2*log(runif(1))
      }   #elseif17 ends
```

```
      } #elseif13 ends



   ### CASE 4
   else if ( t3 == min(ta,t1,t2,t3))   #elseif18 starts    # Case 4
   {
        t <- t3
       C3 <- C3 + 1                                  # number of customers served by 3
       DNd[SS[4]] <- t                       # the departure time for SS[4]

        if (SS[1]==1)   #if8 starts within elseif18       # one customer in SS
         {
            t3 <- 99999
            SS <- c(0,0,0,0)
            }     #if8 ends

       else if (SS[1]==2) #elseif19 starts within elseif18                              #              Two
customer
         {
            t3 <- 99999

            if (SS[3] == 0) #if9 starts within elseif19
             {
                SS <- c(1,SS[2],0,0)
                 }    #if9 ends
```

```
else if ( SS[2] == 0)     #elseif20 starts within elseif19
                 {
                  SS <- c(1,0,SS[3],0)
                   }     #elseif20 ends


          }    #elseif18 ends


   else if (SS[1]==3)    #elseif21 starts within elseif18                    #         Three
Customer
      {
          t3 <-99999
          SS <- c(2,SS[2],SS[3],0)
          }     #elseif21 ends


   else if(SS[1] >3)   #elseif22 starts within elseif18                    #   more   than
three customer
       {
          #m <- max(SS[2],SS[3],SS[4])
          Ns <- next_customer(SS[2], SS[3], SS[4], DNd, Est)
          print(Ns)
          cat("\n")
          SS <- c((SS[1]-1), SS[2], SS[3], Ns)
          t3 <- t-1/r3*log(runif(1))
          }      #elseif22 ends


   }     #elseif18 ends
```

```
   } #repeat1 ends

         ave.1[j]<- mean(DNd-ANa)

  }    #for-loop1 ends

  mean(ave.1)

  Nsl<-Ns

  replica1[i] <- mean(ave.1)
```

Situation THREE: each teller has their own line

```
n.rep <- 1000
ave.3 <- numeric(n.rep)

for (j in 1:n.rep)
 {

 ### Main Program ###
 ## Initialization ##
 t <- 0              # current time
 T <- 9              # after T, no one can enter this system
 Na <- 0                      # total number of arrival
 S1 <- S2 <- S3 <- numeric()   # Si is the sequence of customers in the server i at time t
 Nd <- 0                      # total number of departure
```

```
T0 <- numeric(1)
r1 <- 6          # the service time for server i has exponential distn with rate ri
r2 <- 5
r3 <- 4

t <- t- (1/lambda_t(t)$lamt)*log(runif(1))     # generate the first arrival time
ta <- T0 <- t              # next arrival
t1 <- t2 <- t3 <- 99999         # ti is the service completion time of the customer
                                # presently being served by server i
ANa <- numeric()      # the arrival time of customer n
DNd <- numeric()      # the departure time of customer n
Est <- numeric()  #Estimated time of service at the time of arrival

repeat {
    if(ta==99999 && ta==t1 && t1==t2 && t2==t3)
            break

        ### CASE 1

    if (ta == min(ta,t1,t2,t3))       # Case 1
     {
        t <- ta
        Na <- Na + 1
        ta <- generate_Tt(ta,t1,t2,t3,t)$ta  # new arrival time
        ANa[Na] <- t
        Est[Na] <- runif(1)*10
```

```
if (length(S1) == min(length(S1),length(S2),length(S3)))      # enter 1
  {
        Ns <- next_customer(S1, S2, S3, DNd, Est)
        #print(Ns)
        #cat("\n")

        S1[length(S1)+1] <- Ns          # customer Ns enter the server 1

        if (length(S1) == 1)
         {
                t1 <- t - (1/r1)*log(runif(1))     # the time of server being completed
                }
        }

else if (length(S2) == min(length(S1),length(S2),length(S3)))          # enter 2
        {
         Ns <- next_customer(S1, S2, S3, DNd, Est)
         #print(Ns)
         #cat("\n")
         S2[length(S2)+1] <- Ns          # customer Ns enter the sever 2

         if (length(S2) == 1)
          {
                t2 <- t - (1/r2)*log(runif(1))     # the time of server being completed
                  }
          }

else if (length(S3) == min(length(S1),length(S2),length(S3)))          # enter 2
```

```
                {
                 Ns <- next_customer(S1, S2, S3, DNd, Est)
                 #print(Ns)
                 #cat("\n")
                 S3[length(S3)+1] <- Ns          # customer Ns enter the sever 3

                 if (length(S3) == 1)
                         {
                          t3 <- t - (1/r3)*log(runif(1))    # the time of server being completed
                           }
            }

    }




### CASE 2

        else if ( t1 == min(ta,t1,t2,t3))  # Case 2
      {
                t <- t1
                DNd[S1[1]] <- t1         # The first in server will leave
                S1 <- S1[-1]     # the first leave

                if (length(S1) == 0)
                        t1 <- 99999

                if (length(S1) > 0)
```

```
                t1 <- t - (1/r1)*log(runif(1))     # the time of server being completed
        }



### CASE 3
      else if ( t2 == min(ta,t1,t2,t3))  # Case 3
       {
            t <- t2
            DNd[S2[1]] <- t2         # The first in server will leave
            S2 <- S2[-1]     # the first leave

            if (length(S2) == 0)
                t2 <- 99999

            if (length(S2) > 0)
                t2 <- t - (1/r2)*log(runif(1))     # the time of server being completed
      }

### CASE 4
  else if ( t3 == min(ta,t1,t2,t3))  # Case 4
   {
            t <- t3
            DNd[S3[1]] <- t3         # The first in server will leave
            S3 <- S3[-1]     # the first leave
            if (length(S3) == 0)
                t3 <- 99999
            if (length(S3) > 0)
                t3 <- t - (1/r3)*log(runif(1))     # the time of server being completed
```

```
                    }
  }      #repeat ends
 ave.3[j] <- mean(DNd-ANa)
 }   #for-loop ends

Nsl
mean(ave.1)
Ns
mean(ave.3)

replica3[i] <- mean(ave.3)
}
```

Approach3:

```
## subroutine ##
## intensity rate in which the customers arrive ##
lambda_t <- function(time)      # the banks open from 8am to 5pm, so we set
        {
                if(time >= 4 && time <= 5)     # the time from 0 to 9 and the rate is 4/hr
                        {
                                lambda <- 6                  # but increase to 6 between 4 and 5.
                                }
                else
                 {
                        lambda <- 4
                        }

                list(lamt=lambda)
                }

### subroutine ###
### Generate Tt which will be assigned to ta: next arrival time ###

generate_Tt <- function(ta,t1,t2,t3,t)
        {


                Tt <- t - (1/lambda_t(t)$lamt)*log(runif(1))    # next arrival time


                if (ta < 4)                    # ta is less than 4
                {
                        ta <- Tt
```

```
                if (ta > 4)      #if greater than 4, change rate
                        {
        ta <- 4+((ta-4)*4/6)
         }
            }

            else if ( 4 <= ta && ta <= 5 )
                    {
                            # 4 <= ta <= 5
                            ta <- Tt
                            if (ta > 5)
                             {
                                    ta <- 5+((ta-5)*6/4)        #if greater than 5 , change rate
                                    }
                            }
            else
                    ta <- Tt

            if ( ta > 9)
                    ta <- 99999      # To insure that nobody can enter the system after 5 pm

            list(ta=ta)
    }
```

###Subroutine to determine the next customer to be served in the system from the current waiting line(s)

##It takes inputs: customers currently in the system being served, latest departed list of customers,

#latest arrival list of customers

##It returns the next customer based on the waiting time

```
 next_customer <- function(s1, s2, s3, depart, arriv)
 {
	ins <- union(s1, union(s2, s3))
	ins_depart <- union(ins, which(depart!=FALSE))
	in_line <- setdiff(which(arriv!=FALSE), ins_depart)
	next_cust <- which(arriv == max(arriv[in_line]))

	next_cust = next_cust
 }

replica1 <- numeric()
replica3 <- numeric()

for (i in 1:25)
{

### Situation ONE: One line with parallel servers
n.rep <- 1000
ave.1 <- numeric(n.rep)

for (j in 1:n.rep)
 {
	## Main Program ##
	## Initialization ##
	t <- 0		# current time
	T <- 9		# after T, no one can enter this system
	n <- 0		# number of people is in the system in the time t
	Na <- 0			# total number of arrival
```

```
C1 <- C2 <- C3 <- 0    # Ci is the number of customers served by i by time t
Nd <- 0                # total number of departure
T0 <- numeric(1)
r1 <- 6          # the service time for server i has exponential distn with rate ri
r2 <- 5
r3 <- 4

lambda <- 6
SS <- c(0,0,0,0)       # SS(n,i1,i2,i3) n:number of customers in the system
                            # ij: person i is being served by server j

t <- t- (1/lambda_t(t)$lamt)*log(runif(1))      # generate the first arrival time

ta <- T0 <- t          # next arrival
t1 <- t2 <- t3 <- 99999        # ti is the service completion time of the customer,
        #presently being served by server i

ANa <- numeric()       # the arrival time of customer n
DNd <- numeric()       # the departure time of customer n
Est <- numeric()       #estimated time of service at the time of arrival

repeat #repeat starts
{
    if(ta==99999 && ta==t1 && t1==t2 && t2==t3)   #if0 starts
            break       #if0 ends  #Abnormal termination

    ### CASE 1
```

```
if (ta == min(ta,t1,t2,t3))        #if1 starts within repeat  #Case 1
 {
        t <- ta
        Na <- Na + 1
ANa[Na] <- t

        meanr <- mean(r1, r2, r3)
    #Est[Na] <- runif(1)*10

        ta <- generate_Tt(ta,t1,t2,t3,t)$ta        # NOtice: the change of rate

        if (SS[1] == 0)  #if2 starts within if1                    # nobody is in the system
         {
                Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
                #print(Ns)
                #cat("\n")
                SS <- c(1,Ns,0,0)                   # go to server 1 if all are free
                t1 <- t - (1/r1)*log(runif(1))     # the time of service being completed
                }   #if2 ends

        else if (SS[1] == 1)     #elseif1 starts within if1          # only one is in the system
         {
                 SS[1] <- 2
                 if(SS[3]==0 && SS[4]==0) #if2 starts within elseif1
                         {
```

```
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
#print(Ns)
#cat("\n")
SS[3] <- Ns
t2 <- t-(1/r2)*log(runif(1))
}   #if2 ends

else if (SS[2] == 0 && SS[4] == 0) #elseif2 of if2 within elseif1
{
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
#print(Ns)
#cat("\n")
SS[2] <- Ns
t1 <- t-(1/r1)*log(runif(1))
}     #elseif2 ends

else if (SS[2] == 0 && SS[3] == 0) #elseif3 of if2 within elseif1
{
Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
#print(Ns)
#cat("\n")
SS[2] <- Ns
t1 <- t-(1/r1)*log(runif(1))
}      #elseif3 ends
}   #elseif1 ends
```

```
else if (SS[1] == 2)    #elseif4 starts within if1        # two customers are in the system
{
    SS[1] <- 3
    if(SS[4]==0) #if3 starts within elseif4
        {        Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
                 #print(Ns)
        #cat("\n")
                 SS[4] <- Ns
                 t3 <- t-(1/r3)*log(runif(1))
                 }    #if3 ends

    else if (SS[3] == 0)   #elseif5 starts within elseif4
        {
    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
        #print(Ns)
    #cat("\n")
        SS[3] <- Ns
        t2 <- t-(1/r2)*log(runif(1))
        }     #elseif5 ends

    else if (SS[2] == 0)    #elseif6 starts within elseif4
        {
    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa) #next customer who has been waiting longest
        #print(Ns)
    #cat("\n")
```

```
                    SS[2] <- Ns
                    t1 <- t-(1/r1)*log(runif(1))
                    }     #elseif6 ends

                }       #elseif4 ends



            else if (SS[1] > 2)      #elseif7 starts within if1                    # more than 2
customers
                {
                    SS[1] <- SS[1]+1
                    }    #elseif7 ends

        }    #if1 ends

### CASE 2
    else if ( t1 == min(ta,t1,t2,t3))  #elseif8 starts  #Case 2
    {
        t <- t1
        C1 <- C1 + 1                          # number of customers served by 1
        DNd[SS[2]] <- t                       # the departure time for SS[2]



        if (SS[1]==1)  #if4 starts  within elseif8                      # one customer in SS
          {
            t1 <- 99999

            SS <- c(0,0,0,0)
      }        #if4 ends
```

```
else if (SS[1]==2) #elseif9 starts within elseif8 # Two customer
{
    t1 <- 99999
    if (SS[4] == 0) #if5 starts within elseif9
     {
        SS <- c(1,0,SS[3],0)
        }    #if5 ends

    else if ( SS[3] == 0)  #elseif10 starts within elseif9
       {
         SS <- c(1,0,0,SS[4])
     }   #elseif10 ends

     }  #elseif9 ends

else if (SS[1]==3) #elseif11 starts within elseif8 # Three Customer
{
    t1 <-99999
    SS <- c(2,0,SS[3],SS[4])
    }      #elseif11 ends

else if(SS[1] >3) #elseif12 starts within elseif8 # more than three customer
 {
    #m <- max(SS[2],SS[3],SS[4])
    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
```

```
                    #print(Ns)
                    #cat("\n")
                    SS <- c((SS[1]-1), Ns, SS[3], SS[4])
                    t1 <- t-1/r1*log(runif(1))
                    }    #elseif12 ends

            }      #elseif8 ends

    ### CASE 3
    else if (t2 == min(ta,t1,t2,t3)) #elseif13 starts # Case 3
                {
                    t <- t2
                    C2 <- C2 + 1                        # number of customers served by 2
                    DNd[SS[3]] <- t                     # the departure time for SS[3]

                    if (SS[1]==1) #if6 starts within elseif13           # one customer in SS
                    {
                        t2 <- 99999
                        SS <- c(0,0,0,0)
                    }     #if6 ends

                    else if (SS[1]==2) #elseif14 of if6 starts within elseif13
# Two customer
                     {
                        t2 <- 99999
                        if (SS[4] == 0) #if7 starts within elseif14
                         {
                            SS <- c(1,SS[2],0,0)
                            } #if7 ends
```

```
else if ( SS[2] == 0)    #elseif15 of if7 starts within elseif14
  {
        SS <- c(1,0,0,SS[4])
  }   #elseif15 ends


   }     #elseif14 ends


else if (SS[1]==3)  #elseif16 starts within elseif13      # Three Customer
 {
      t2 <-99999
      SS <- c(2,SS[2],0,SS[4])
      }     #elseif16 ends


else if(SS[1] >3)   #elseif17 starts within elseif13                          # more
than three customer
 {
      #m <- max(SS[2],SS[3],SS[4])
      Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
      print(Ns)
      cat("\n")
      SS <- c((SS[1]-1), SS[2], Ns, SS[4])
      t2 <- t-1/r2*log(runif(1))
      }   #elseif17 ends


} #elseif13 ends
```

```
### CASE 4
else if ( t3 == min(ta,t1,t2,t3))   #elseif18 starts    # Case 4
{
    t <- t3
    C3 <- C3 + 1                              # number of customers served by 3
    DNd[SS[4]] <- t                   # the departure time for SS[4]

    if (SS[1]==1)   #if8 starts within elseif18      # one customer in SS
     {
        t3 <- 99999
        SS <- c(0,0,0,0)
        }    #if8 ends

    else if (SS[1]==2) #elseif19 starts within elseif18                    #          Two
customer
     {
        t3 <- 99999

        if (SS[3] == 0) #if9 starts within elseif19
         {
            SS <- c(1,SS[2],0,0)
            }   #if9 ends

   else if ( SS[2] == 0)     #elseif20 starts within elseif19
            {
            SS <- c(1,0,SS[3],0)
             }     #elseif20 ends
```

```
                    }    #elseif18 ends

          else if (SS[1]==3)    #elseif21 starts within elseif18                    #           Three
Customer
      {
                    t3 <-99999
                    SS <- c(2,SS[2],SS[3],0)
                    }    #elseif21 ends

          else if(SS[1] >3)   #elseif22 starts within elseif18                      #   more   than
three customer
          {
                    #m <- max(SS[2],SS[3],SS[4])
                    Ns <- next_customer(SS[2], SS[3], SS[4], DNd, ANa)
                    print(Ns)
                    cat("\n")
                    SS <- c((SS[1]-1), SS[2], SS[3], Ns)
                    t3 <- t-1/r3*log(runif(1))
                    }      #elseif22 ends

      }     #elseif18 ends

  } #repeat1 ends

     ave.1[j]<- mean(DNd-ANa)

 }    #for-loop1 ends
```

```
  mean(ave.1)
  Nsl<-Ns
  replica1[i] <- mean(ave.1)

### Situation THREE: each teller has their own line
n.rep <- 1000
ave.3 <- numeric(n.rep)

for (j in 1:n.rep)
 {

 ### Main Program ###
 ## Initialization ##
 t <- 0                # current time
 T <- 9                # after T, no one can enter this system
 Na <- 0                         # total number of arrival
 S1 <- S2 <- S3 <- numeric()   # Si is the sequence of customers in the server i at time t
 Nd <- 0                         # total number of departure

 T0 <- numeric(1)
 r1 <- 6               # the service time for server i has exponential distn with rate ri
 r2 <- 5
 r3 <- 4

 t <- t- (1/lambda_t(t)$lamt)*log(runif(1))      # generate the first arrival time
 ta <- T0 <- t                   # next arrival
 t1 <- t2 <- t3 <- 99999            # ti is the service completion time of the customer
                                                  # presently being served by server i
```

```
ANa <- numeric()      # the arrival time of customer n
DNd <- numeric()      # the departure time of customer n
Est <- numeric()  #Estimated time of service at the time of arrival

repeat {
        if(ta==99999 && ta==t1 && t1==t2 && t2==t3)
                        break

                ### CASE 1

        if (ta == min(ta,t1,t2,t3))        # Case 1
          {
                t <- ta
                Na <- Na + 1
                ta <- generate_Tt(ta,t1,t2,t3,t)$ta  # new arrival time
                ANa[Na] <- t
                Est[Na] <- runif(1)*10

        if (length(S1) == min(length(S1),length(S2),length(S3)))      # enter 1
           {
                Ns <- next_customer(S1, S2, S3, DNd, Est)
                #print(Ns)
                #cat("\n")

                S1[length(S1)+1] <- Ns          # customer Ns enter the server 1

                if (length(S1) == 1)
                  {
```

```
            t1 <- t - (1/r1)*log(runif(1))    # the time of server being completed
            }
        }

else if (length(S2) == min(length(S1),length(S2),length(S3)))         # enter 2
        {
         Ns <- next_customer(S1, S2, S3, DNd, Est)
         #print(Ns)
         #cat("\n")
         S2[length(S2)+1] <- Ns        # customer Ns enter the sever 2

         if (length(S2) == 1)
          {
            t2 <- t - (1/r2)*log(runif(1))    # the time of server being completed
             }
          }

else if (length(S3) == min(length(S1),length(S2),length(S3)))         # enter 2
        {
         Ns <- next_customer(S1, S2, S3, DNd, Est)
         #print(Ns)
         #cat("\n")
         S3[length(S3)+1] <- Ns        # customer Ns enter the sever 3

         if (length(S3) == 1)
            {
             t3 <- t - (1/r3)*log(runif(1))   # the time of server being completed
             }
```

```
        }

    }



### CASE 2

        else if ( t1 == min(ta,t1,t2,t3))  # Case 2
    {
                t <- t1
                DNd[S1[1]] <- t1        # The first in server will leave
                S1 <- S1[-1]     # the first leave

                if (length(S1) == 0)
                        t1 <- 99999

                if (length(S1) > 0)
                        t1 <- t - (1/r1)*log(runif(1))     # the time of server being completed
           }



### CASE 3
        else if ( t2 == min(ta,t1,t2,t3))  # Case 3
         {
                t <- t2
                DNd[S2[1]] <- t2        # The first in server will leave
                S2 <- S2[-1]     # the first leave
```

```
                if (length(S2) == 0)
                        t2 <- 99999

                if (length(S2) > 0)
                        t2 <- t - (1/r2)*log(runif(1))     # the time of server being completed
        }



### CASE 4
  else if ( t3 == min(ta,t1,t2,t3))  # Case 4
   {
                t <- t3
                DNd[S3[1]] <- t3         # The first in server will leave
                S3 <- S3[-1]     # the first leave

                if (length(S3) == 0)
                        t3 <- 99999

                if (length(S3) > 0)
                        t3 <- t - (1/r3)*log(runif(1))     # the time of server being completed
                }

  }     #repeat ends

ave.3[j] <- mean(DNd-ANa)

 }   #for-loop ends
```

```
Nsl
mean(ave.1)
Ns
mean(ave.3)

replica3[i] <- mean(ave.3)

}
```

Part2

```
## subroutine ##
## intensity rate in which the customers arrive ##
lambda_t <- function(time)     # the banks open from 8am to 5pm, so we set
        {
                if(time >= 4 && time <= 5)    # the time from 0 to 9 and the rate is 4/hr
                        {
                                lambda <- 6                # but increase to 6 between 4 and 5.
                                }
                else
                 {
                        lambda <- 4
                        }
                list(lamt=lambda)
                }
### subroutine ###
### Generate Tt which will be assigned to ta: next arrival time ###
generate_Tt <- function(ta,t1,t2,t3,t)
        {
                Tt <- t - (1/lambda_t(t)$lamt)*log(runif(1))    # next arrival time

                if (ta < 4)                  # ta is less than 4
                {
                        ta <- Tt
                        if (ta > 4)      #if greater than 4, change rate
                                {
            ta <- 4+((ta-4)*4/6)
             }
```

```
                }

                else if ( 4 <= ta && ta <= 5 )
                        {
                                # 4 <= ta <= 5
                                ta <- Tt
                                if (ta > 5)
                                 {
                                        ta <- 5+((ta-5)*6/4)        #if greater than 5 , change rate
                                        }
                                }
                else
                        ta <- Tt

                if ( ta > 9)
                        ta <- 99999      # To insure that nobody can enter the system after 5 pm

                list(ta=ta)
          }
### Situation TWO : Tandem system
replica <- numeric()
for (i in 1:25){
n.rep <- 50000
ave.2 <- numeric(n.rep)
for (j in 1:n.rep)    #for-loop2 starts
 {
 ### Main Program ###
 ## Initialization ##
```

```
t <- 0                    # current time
Na <- 0                         # total number of arrival
Nd <- 0                         # total number of departure
SS <- c(0,0,0) # SS(n1,n2,n3) ni:number of customers in the server i
n1 <- n2 <- n3 <- 0
r1 <- 4        #Teller 1 becomes the slowest
r2 <- 6        #Teller 2 becomes the slowest when we swap r1 with r2
r3 <- 5        #Teller 3 becomes the slowest when we swap r1 with r3

t <- t- (1/lambda_t(t)$lamt)*log(runif(1))     # generate the first arrival time
ta <- T0 <- t             # next arrival
t1 <- t2 <- t3 <- 99999         # ti is the service completion time of the customer
                                      # presently being served by server i

ANa.1 <- numeric()    # the arrival time of customer n
ANa.2 <- numeric()    # the arrival time of customer n at server 2
ANa.3 <- numeric()  # the arrival time of customer n at server 3
ANa <- numeric()
DNd <- numeric()      # the departure time of customer n

repeat        #repeat2 starts
 {
    if(ta==99999 && ta==t1 && t1==t2 && t2==t3) #if10 starts
        break       #if10 ends

    ### CASE 1

    if (ta == min(ta,t1,t2,t3))   #if11 starts          # Case 1
```

```
        {
                t <- ta
                Na <- Na + 1
                n1 <- n1 + 1
                ta <- generate_Tt(ta,t1,t2,t3,t)$ta       # NOtice: the change of rate

                if (n1 == 1)    #if12 starts within if11
                  {
                        t1 <- t - (1/r1)*log(runif(1))    # the time of service being completed
                        }    #if12 ends

                ANa.1[Na] <- t
#               ANa[Na] <- t
     }       #if11 ends

                ### CASE 2
        else if ( t1 == min(ta,t1,t2,t3))   #elseif23 starts # Case 2
        {
                t <- t1
                n1 <- n1 - 1
                n2 <- n2 + 1

                if (n1 == 0)    #if13 starts within else23
                  {
                        t1 <- 99999
                        }

                else
```

```
    {
        t1 <- t -(1/r1)*log(runif(1))
        }

    if (n2 == 1)
    {
        t2 <- t -(1/r2)*log(runif(1))
          }

    ANa.2[(Na-n1)] <- t

 }



    ### CASE 3
else if ( t2 == min(ta,t1,t2,t3))  # Case 3
  {
    t <- t2
    n2 <- n2 - 1
    n3 <- n3 + 1
    if (n2 == 0)
      {
        t2 <- 99999
         }
    else
    {
        t2 <- t -(1/r2)*log(runif(1))
        }
```

```
          if (n3 == 1)
            {
                    t3 <- t -(1/r3)*log(runif(1))
                     }

          ANa.3[(Na-n1-n2)] <- t
   }




          ### CASE 4
else if ( t3 == min(ta,t1,t2,t3))  # Case 4
   {
          t <- t3
          n3 <- n3 - 1
          Nd <- Nd + 1

          if (n3 == 0)
            {
                    t3 <- 99999
                     }

          else
           {
                    t3 <- t -(1/r3)*log(runif(1))
                    }
```

```
            DNd[(Nd)] <- t
        }

    }     #repeat ends

 ave.2[j] <- mean(DNd-ANa.1)
 }   #for-loop ends

#mean(ave.2)
replica[i] <- mean(ave.2)
 }
```