Operating Systems-II
Report-Asgn-4

Task:

The goal of this assignment is to implement Korean Restaurant Problem using Synchronisation.

Approach and Implementation:

1. To solve this problem I have used the semaphores as given in the sample solution, it is as follows,

```
eating = 0
waiting = 0 // No. of threads eating and waiting at table.

mutex = Semaphore (1)
block = Semaphore (0) //incoming customers wait on block
must_wait = FALSE // indicates that the table is full

mutex . wait ()

if ( must_wait == TRUE || eating+1 > X )
      waiting++
      must_wait = TRUE
      mutex . signal ()
      block . wait ()
else
      eating++
      must_wait = ( waiting>0 && eating == X)
      mutex . signal ()

# Customer eats at the table

mutex . wait ()
eating--
if eating == 0
      k = min (X , waiting )
      waiting -= k
      eating += k
      must_wait = ( eating == X)
      block . signal (k)
mutex . signal ()
```

2. To log the info of the threads I have used a multimap<string, string> ordered by ascending wrt time a thread enters and updated it only when the thread has the lock so there won't be any race condition.

3. For getting the current exact time I have written a function called getSysTime() which returns a string representing the exact system time in HH:MM:SS:millisec:musec format so that it would be clear to analyze race conditions.

```
string getSysTime() { // getSystem Time
 char buf[26];
 time(&t);
 struct tm *time_info;
 struct timeval tv;
 gettimeofday(&tv, NULL);
 time_info = localtime(&t);
 int ms = tv.tv_usec / 1000, musec = tv.tv_usec % 1000;
 sprintf(buf, "%.2d:%.2d:%.2d:%.3d:%.3d", time_info->tm_hour,
         time_info->tm_min, time_info->tm_sec, ms,
         musec); // writing time to buffer
 string sysTime(buf);
 return sysTime;
}
```

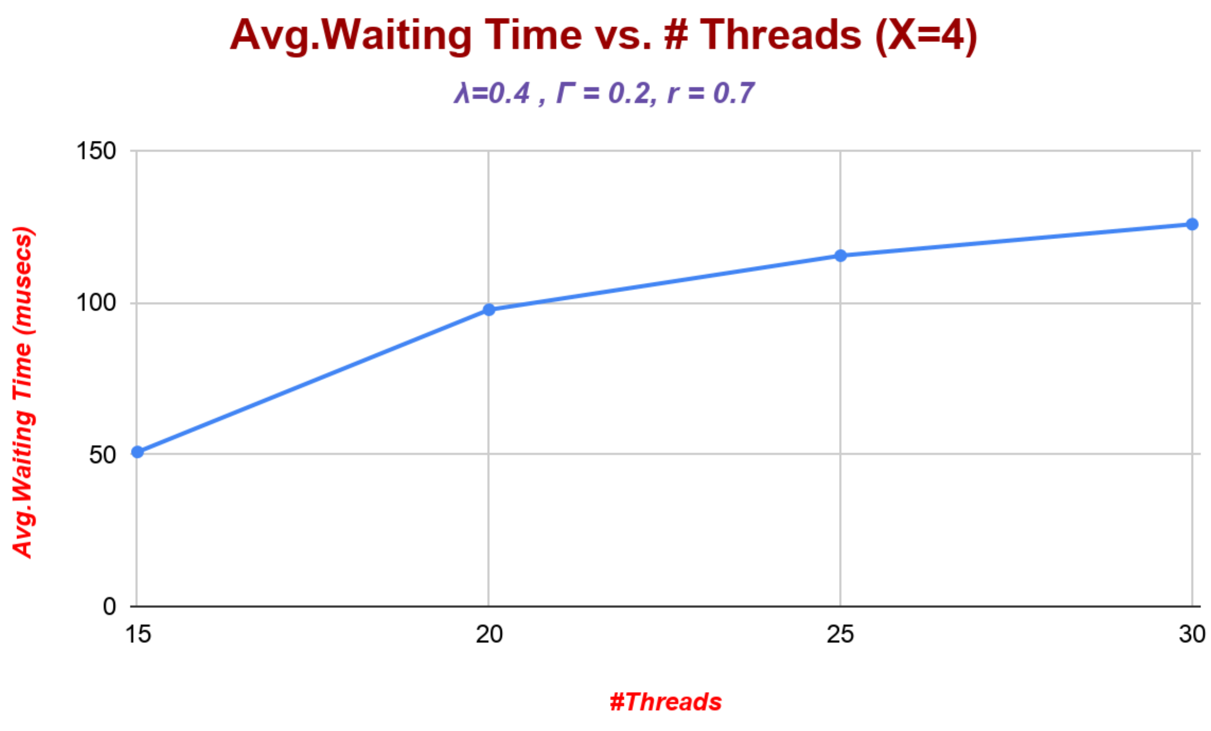
4. As per the requirement of the question, I have used `default_random_engine generator(time(NULL)),chrono::duration<double> uniform_int_distribution<int>, exponential_distribution<double>` To simulate the process as described in the question.

5. As we are already storing the sorted information wrt time in our multimap, I have just printed the logging info by just iterating the multimap.

Graphs Analysis:

Graph #1: Input Params:

N: No. of threads varies from 15 to 30 in steps of 5.
X: Fixed to 4
λ=0.4 , Γ = 0.2, r = 0.7



Why these Parameters?

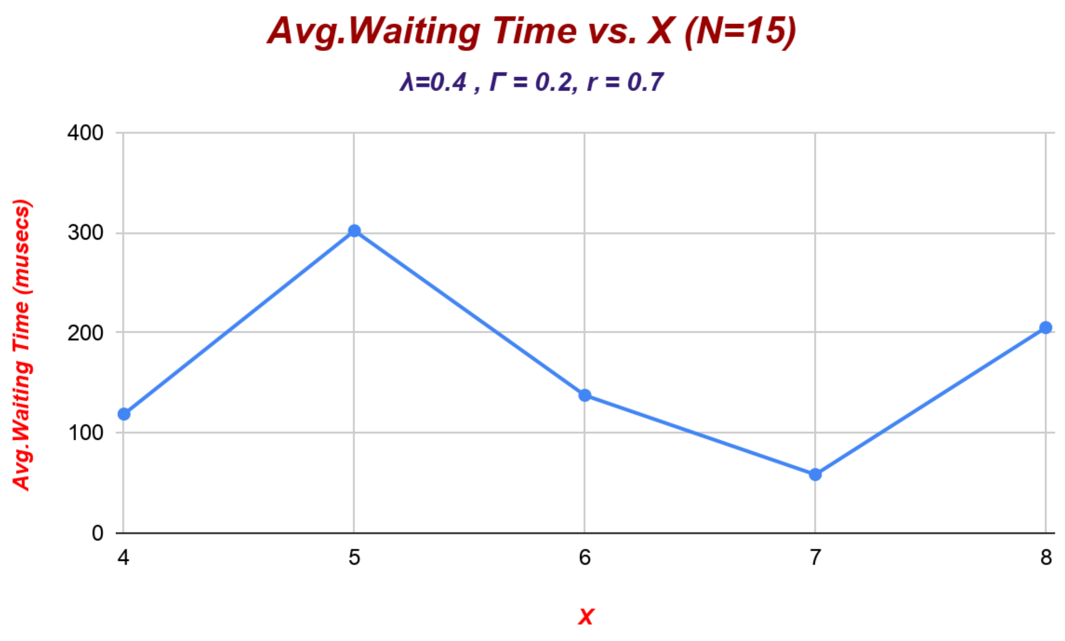
- The value I have taken is λ > Γ, therefore the average waiting time for lambda will be less than the average waiting time for gamma. As people will be arriving at the restaurant in very small intervals and people will take more time eating, the restaurant will be busy most of the time. Hence the customers have to wait for more time. Therefore as N increase, the waiting time should increase.

- We can see that as the number of threads increases the waiting time is increasing.

- The values may differ by running each time as the random number generators may generate various numbers b/w given values.

Graph #2: Input Params:

X: Number of customers per group varies from 4 to 8 in steps of 1
N: Fixed to 15
λ=0.4 , Γ = 0.2, r = 0.7



- Ideally, as the value of X increases, more people can get a seat in the restaurant. therefore the waiting time should decrease.

- We can see that there is an increase and then a decrease in average waiting times as the X value is increasing.

- The values may differ by running each time as the random number generators may generate various numbers b/w given values.