# Assignment 6

**Jai Arora**

**2018CS50219**

**Details :** Most of the implementation details are documented in the code, but here are the main points-

1. The teller queue is a simple queue, where the object added goes at the last, while the removal is carried out from the front. It is implemented as a queue of customers
2. The event queue is a linear priority queue, which is always sorted in order of the event arrival times. It is implemented as a priority queue of event objects.
3. The event object is of 4 types :- New Customer event, served customer event, teller idle event, and a serving teller event.
4. The statistics which are to be calculated about the simulations are kept as global variables, and are reinitialised to zero whenever a new simulation starts.
5. Each event has a function pointer, which points to its action.
6. In the "One queue per teller" regime, when a customer arrives, and it has to choose a random shortest line, for that I collect all the shortest lines, and choose a random one out of them.
7. Appropriate functions are made to initialise teller queues, customer events and teller events.
8. The tellers are assumed to be in an idle state from the start. Initially they are given a random idle time between 1-600 seconds, and later given a random idle time between 1-150 seconds.
9. A counter is kept to store how many times a function pointer was used in a simulation, and is printed in the end with the statistics.
10. For plotting a curve between average time spent in bank versus number of tellers in the "Common Queue" regime, I range the number of tellers from 1 to 100, and plot the average time spent.
11. A proper directory structure is maintained, and a makefile is made accordingly.

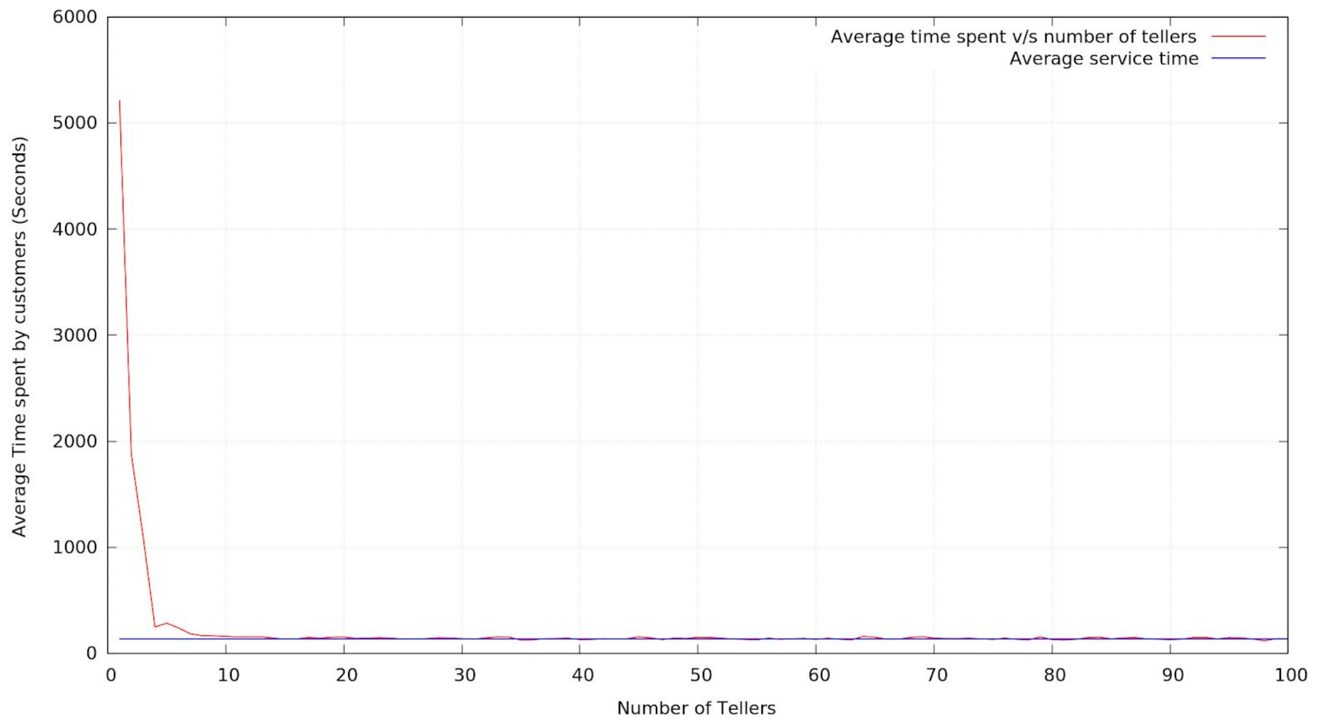
How to run it: The assignment folder contains a makefile. Invoking make in that folder would make an executable named qSim in the folder named "bin". Executing that with proper command line arguments will run the program.

## Test Cases:

1. Command line input: ./bin/qSim 100 4 60 2.3 →

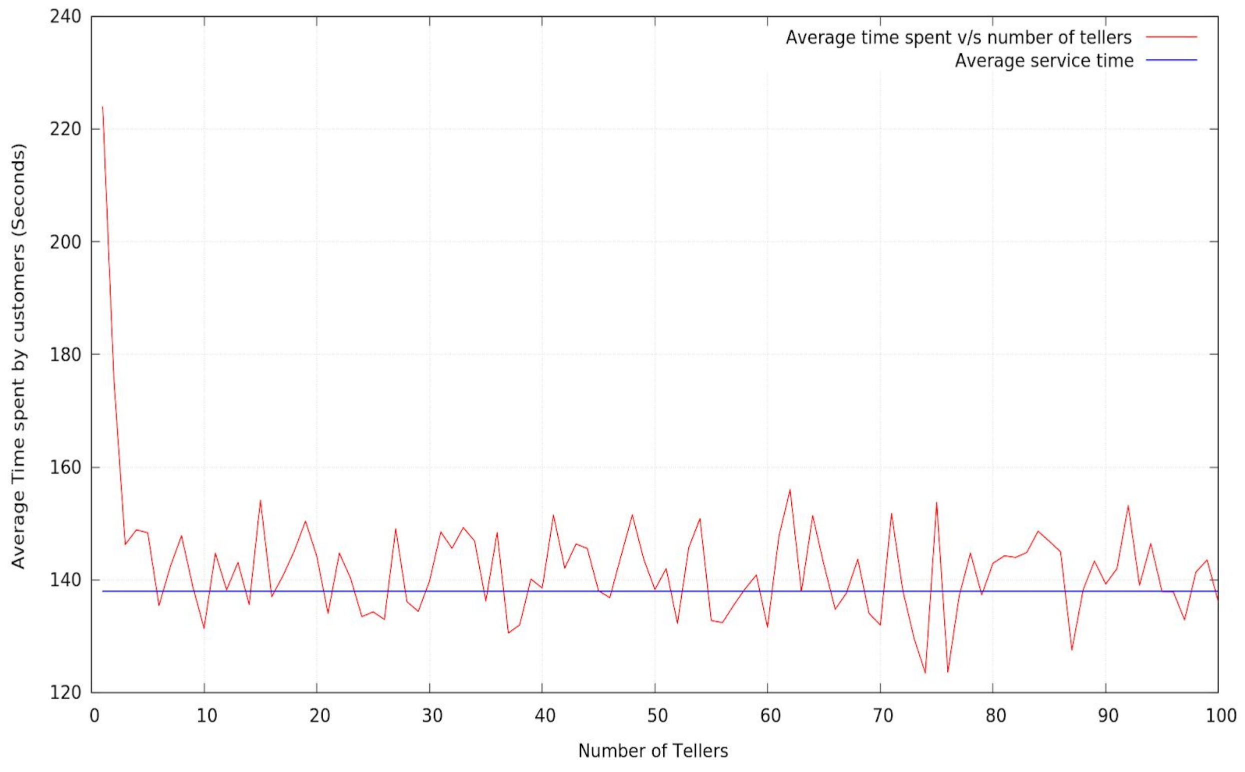
   Output came out to be:

```
Function pointer was used 311 times in this simulation
Total number of customers served: 100
Total Time taken to serve all the customers: 4231.740234 seconds
Number of tellers: 100. Queueing Type : One Queue per Teller
Average amount of time spent by a customer in the bank: 344.906403 seconds
Standard deviation of time spent by the customers: 169.679291 seconds
Maximum Wait Time: 592.451172 seconds
Total teller service time: 14670.274414 seconds
Total teller idle time : 2062.665527 seconds
------------------------------------------------------------------------------------------------------
Function pointer was used 307 times in this simulation
Total number of customers served: 100
Total Time taken to serve all the customers: 4139.063477 seconds
Number of tellers: 4. Queueing Type : Common Queue
Average amount of time spent by a customer in the bank: 411.736023 seconds
Standard deviation of time spent by the customers: 109.130005 seconds
Maximum Wait Time: 493.527344 seconds
Total teller service time: 14596.982422 seconds
Total teller idle time : 1801.647461 seconds
```



2. Command line input: ./bin/qSim 100 4 600 2.3 →
   Output came out to be:

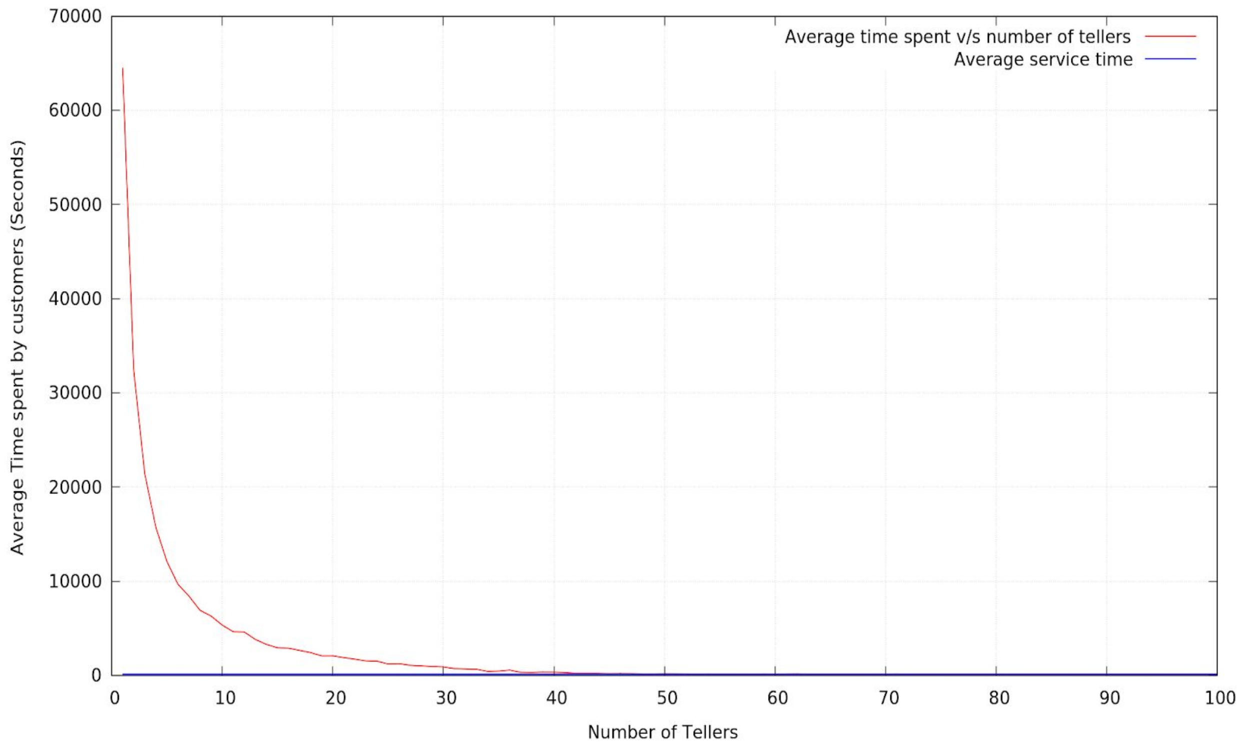
```
Function pointer was used 2023 times in this simulation
Total number of customers served: 100
Total Time taken to serve all the customers: 36071.535156 seconds
Number of tellers: 100. Queueing Type : One Queue per Teller
Average amount of time spent by a customer in the bank: 159.411453 seconds
Standard deviation of time spent by the customers: 77.604507 seconds
Maximum Wait Time: 118.863281 seconds
Total teller service time: 13999.766602 seconds
Total teller idle time : 130214.257812 seconds
---------------------------------------------------------------------------------------------------
Function pointer was used 2014 times in this simulation
Total number of customers served: 100
Total Time taken to serve all the customers: 35966.273438 seconds
Number of tellers: 4. Queueing Type : Common Queue
Average amount of time spent by a customer in the bank: 163.978714 seconds
Standard deviation of time spent by the customers: 88.586052 seconds
Maximum Wait Time: 88.131836 seconds
Total teller service time: 14350.834961 seconds
Total teller idle time : 129451.648438 seconds
```



3. Command line input: ./bin/qSim 1000 4 60 2.3

   Output came out to be:

```
Function pointer was used 3010 times in this simulation
Total number of customers served: 1000
Total Time taken to serve all the customers: 34602.484375 seconds
Number of tellers: 1000. Queueing Type : One Queue per Teller
Average amount of time spent by a customer in the bank: 15746.034180 seconds
Standard deviation of time spent by the customers: 8884.028320 seconds
Maximum Wait Time: 30792.349609 seconds
Total teller service time: 136529.421875 seconds
Total teller idle time : 1807.681885 seconds
------------------------------------------------------------------------------------------------------
Function pointer was used 3007 times in this simulation
Total number of customers served: 1000
Total Time taken to serve all the customers: 34283.218750 seconds
Number of tellers: 4. Queueing Type : Common Queue
Average amount of time spent by a customer in the bank: 15452.199219 seconds
Standard deviation of time spent by the customers: 8871.755859 seconds
Maximum Wait Time: 30524.367188 seconds
Total teller service time: 135987.031250 seconds
Total teller idle time : 925.469666 seconds
```



In all these 3 cases, not much difference was observed between both the queueing types. The max wait time in the common queue is less than the one queue per teller simulation most of the time, while average time spent in the One per teller type is less than or almost the same as the common queue type.
The only main difference I observed was in the graph I obtained. Here are my observations:

1. The graph of average time spent in the bank versus number of tellers seems to be exponentially decaying, and is asymptotic to some value. That value is also drawn on the same graph for comparison.
2. The asymptotic value was found to be directly proportional to the average service time. This was found out by increasing average service time 10 folds, while keeping all other parameters constant.
3. The asymptotic value was the same irrespective of number of customers or the total simulation time, but increasing them slowed the rate of decay of the exponential.
4. The asymptotic value was found to be exactly the average service time, which is pretty obvious because as the number of tellers increase, less number of customers would have to wait for their service, so their total time spent would approach the average service time.

Note: While calculating total idle time of the tellers, I did not take account of the idle time of those tellers who were in idle state when the simulation completed.