Design Document

1. How many threads are you going to use? Specify the task that you intend each thread to perform.
   There are 5 threads for all clerks and then one each for all the customers.

2. Do the threads work independently? Or is there an overall controller thread?
   All threads work independently. There are no controller threads.

3. How many mutexes are you going to use? Specify the operation that each mutex will guard.
   There are 4 mutex. There are 2 mutex that guard the totalTime and totalCustomer for both business and economy queue. The other 2 mutex guard the queueLength for each queque.

4. Will the main thread be idle? If not, what will it be doing?

5. How are you going to represent customers? what type of data structure will you use?
   I have used a pointer structure that stores the class type, user id, arrival and service time. It also has a next pointer which points to the next node in the linked list.

6. How are you going to ensure that data structures in your program will not be modified concurrently?
   I am using the mutex locks to make sure data structures are not being used concurrently. Any changes to the linked list, the queue length, total time and total customers only happen when mutex are locked. This prevents concurrent modification.

7. How many convars are you going to use? For each convar:
   I am using 4 condvars. 2 are being used to make signal to customer that clerk is available. The other 2 make the clerk wait for the customer.
   (a) Describe the condition that the convar will represent.
   The first condition is whether a clerk is available for the customer.
   The second condition is whether the customer's service period is over.

   (b) Which mutex is associated with the convar? Why?
   isClerkFree[] is associated with quequeMutex[].
   condVarQueue[] is associated with quequeMutex[].

   (c) What operation should be performed once pthread cond wait() has been unblocked and re-acquired the mutex?
   In case of condVarQueue, the customer will leave the queue, decrease the queue length and start the service time.

In case of isClerkFree, the clerk will wait until it customer completes service time and sends a signal. Once it receives the signal, it means the clerks job is finished.

8. Briefly sketch the overall algorithm you will use.
   (a) Initially all clerks and customers are assigned one thread each. The customers threads are joined, and program waits till all customer threads are finished.
   (b) All customers will be added to a queue once there wait time is finished.
   Once added to the queue, customers wait. If a clerk signals then they for two conditions:
      i. Customer is the head of the queue.
      ii. No winner is selected

   If true customer then dequeues and starts to perform the service time. Also it calculates the time taken since it entered the queue to the time clerk served him. It uses the waitTimeMutex to make sure the totalTime and totalCustomer variables are available.

   (c) The clerk has an infinite loop. It first checks if either of the two queues are idle. If so it then checks first for the business queue. If there is no one in business customers then it goes to economy queue. If any of these queue have any customer, it locks queue using the queueMutex[i]. It then broadcasts to all customers that it is available. It then passes its clerk id to customer. It set foundWinner[] to false, to indicate it has not found a customer yet. Once an appropriate customer responds, customer will set foundWinner[] to false to indicate clerk has found its customer. Clerk now waits using isClerkFree. When customer is finished, it signals clerk clerks loop starts again.