**1. How many threads are you going to use? Specify the task that you intend each thread to perform.**

- If n is the number of threads used for each customer that will be processed, and since there are 5 clerks present each having their own threads, the total number of threads used will be n + 5.

The customer threads simulate the customer arrival times, enqueue customers into their respective queues, and also signal the conditional variable to notify the clerks.

The clerk threads wait for the customers to go into their queues, dequeue customers, and also simulate the serving times.

**2. Do the threads work independently? Or, is there an overall “controller” thread?**

- The customer threads work independently since they enqueue themselves into their respective queues. The clerk threads rely on the signals sent from the customer threads to start processing the customers.

**3. How many mutexes are you going to use? Specify the operation that each mutex will guard.**

- There is an array of 5 mutexes for the 5 clerks. The queue mutexes guard access to the bussinessQueue and economyQueue, ensuring that the customers can be enqueued and dequeued safely.

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

- The main thread will not be idle. It initializes queues, mutexes, and conditional variables. Along with this, it creates customer and clerk threads, and waits for all those threads to finish. It calculates the average waiting times.

**5. How are you going to represent customers? what type of data structure will you use?**

- The customers are represented using a struct called “Customer”, that holds the specific information that is extracted from the input file. They are then sorted into their respective queues based on the class-type of the customer.

**6. How are you going to ensure that data structures in your program will not be modified concurrently?**

- The mutex and condition variable, “mutex” and “conVar” respectively, are used together so that only the required thread and queues are modified at a time.

**7. How many convars are you going to use? For each convar:**

- Only one condition variable, “conVar”, is used.

**(a) Describe the condition that the convar will represent.**

- Indicates whether at least one customer is present in the queue, ready to be served.

**(b) Which mutex is associated with the convar? Why?**

- The “mutex” mutex is associated with the “conVar” to ensure that queues are checked and modified only when needed.

**(c) What operation should be performed once pthread\_cond\_wait() has been unblocked and reacquired the mutex?**

- Once pthread\_cond\_wait() has been unblocked, the clerk thread will dequeue a customer and serve them. It then releases the mutex.

**8. Briefly sketch the overall algorithm you will use. You may use sentences such as: If clerk i finishes service, release clerkImutex.**

- Initialize queues, mutexes, and condition variables. Then the input file is read, and customer and clerk threads are created.

Customer thread:

* Simulate the arrival time and lock the “mutex”.
* Enqueue the customers into their respective queues.
* Signal the “conVar” and then unlock the “mutex”.

Clerk thread:

* Lock the “mutex” and wait for the signal from “conVar”.
* Dequeue customer from their queue and unlock the “mutex”.
* Serve the customer.
* Update the wait time variables.

Join all the customer and clerk threads and calculate the average waiting times. Then destroy the mutexes and condition variables.