**SUMMARY**

Simulation Details

* The postal worker problem seemed to be a common thread synchronization problem at first. However, the finer nuances became apparent to me during an in-depth analysis of this project.
* This project has different synchronization problems when compared to barbershop problem. The problems I faced were mostly during design stage of the project.

Problems Faced

* Synchronizing the threads while the customer places the order and the postal worker receives it was a little challenging. I had to resort to using three different queues , one for each ID otherwise I will have concurrency issues.
* I also could not keep a single mutex across all queues because all queues will get blocked if any is being used which will delay the entire execution. Hence, I kept three queues and three mutex semaphores.
* Also, replication of an error can be very complex if you thread scheduling policies are unknown or contradictory.
* I had to make sure to use the right semaphores to maintain mutual exclusion for access to shared resources. I missed it a couple of times in the beginning and my data structure would become corrupt or fail.

What I learnt?

* I learnt how to share resources between various threads and also the need to lock these resources during access by any thread.
* I learnt that thread synchronization is a complex process and there are a lot of variations in controlling these threads.
* I learnt that using too many mutexes slows down the execution of the program. Hence, they must be kept to a minimum value.
* The most important thing I learnt was the value of designing the code before implementing it. Due to the sheer abundance of semaphores and their complex relations, a poor design can result in a very messy code. Debugging such code is very time consuming and frustrationg.

Results

I was successfully able to simulate the multi-threading system. I could also ensure its efficient execution. I did not use sleep() as a method of synchronization. Instead, I used semaphores only thus making sure that my simulation executed in the least possible time.