ACS Design Document

1. The program uses 4 + n threads, where 4 represents the number of clerks and n represents the number of customers.
2. The threads work independently.
3. The program uses 2 + 4 + 1 = 7 mutexes, where 2 represents the business and economy queues, 4 represents each clerk, and 1 represents a time mutex for finding waiting times to be averaged at the end.
4. The main thread will be idle.
5. Customers are represented using a linked list of nodes which contain all the relevant information regarding a customer (service time, user id, etc.). Queues are used to store user IDs in the correct order that users enter the queues in.
6. Using mutex locks it can be ensured that the linked lists and queues are not modified by other concurrently running threads when they should not be.
7. The program uses 2 + 4 = 6 condvars, where 2 represents the economy and business queues and 4 represents the clerks.
    a. The condition that the queue condvars represent is: “Is the clerk ready to serve a customer?” and the condition the clerk condvars represent is: “Is the clerk done serving the customer?”
    b. The queue condvar and the clerk condvars are represented by the queue and clerk mutexes respectfully. This is because when a thread is waiting on each condvar, we need to unlock the mutex that the thread had locked so other threads waiting on these locks can perform tasks.
    c. Unlock the mutex that it has reacquired.
8. Pseudo-code:

    Initialize all global variables (mutex, condvar, etc)
    Create clerks
        While no customers in queues, do nothing
        If customer enters queue
            Pick business if business is not full
            Else pick economy if it is not full
        Lock queue mutex, Clerk i signals to selected queue that it is ready for service, unlock queue mutex
        Lock clerk mutex, Clerk i waits for service to finish, unlock clerk mutex
    Create customers
        Customer j arrives
        Lock queue mutex, Enqueue(j) into selected queue
        Customer j waits for clerk to signal it
        Dequeue(j), unlock queue mutex
        Once customer j is signaled it begins service
        After service customer j signals clerk i that service is complete