1. For the program, 6 threads will be created, where one is for process the customer queue and the rest 5 threads are for each clerks.

   The customer queue thread enqueue the arriving customer at current time and wait for the next one.

   The clerk thread will dequeue the customer, record and update the total waiting time for customers.

2. The customer thread will have a conditional variable to have the overall control, preventing clerk thread from dequeuing an empty queue when there is still customer not arrived which can cause segmentation fault for the linked list design in the program.

3. One mutex will be used. It will prevent clerk threads from dequeuing concurrently.

4. In the program, customer queue is the main thread controlling others. It will enqueue customers and wait clerk to process customer until all the customers are either on queue. Then it will exit.

5. Link list with each node recording each customer's info will be the data structure

6. The customer thread has a conditional variable to ensure the linked list not being dequeued by the clerk concurrently. Also, as soon as the clerk thread get the convar wait signal, it will lock the mutex and perform operation on the sensitive variable or data structure.

7. One convar will be used:

   a) The condition variable represents if there is any clerk at current time is available.
   b) The mutex preventing clerk threads from dequeuing concurrently. The reason is all clerk can also modify the waiting queue. Associating the mutex prevent the data being edited concurrently.
   c) After clerk thread signals, the customer queue thread first check if there is any available clerk currently and assign customer if there is any. Then, the customer queue thread will wait for next.

8.

```
Main(){
        Prompt user to input file name
        process_file(filename)
        simulate()
}
```

```
process_file(filename){

        format and store file data
        check if data is legal
        store number of customer on each class

}

Simulate(){
        Create customer_entry() thread
        Create 5 clerk() thread
        Calculate waiting time from global value
        Join and destroy threads
}

customer_entry() {
        lock_mutex()
        While(there is customer on queue or incoming){
                Update time and wait the next customer
                Put the arrived customer in appropriate queue based on class
                Output the info

                Check the availability of clerks

                If(any clerk is free){
                        Pthread_cond_wait(&condvar, &lock);
                }
        }
        unlock_mutex()
        pthread_exit();
}

clerk(){
        lock_mutex()
        while(not all customer are serviced){
                if(business class queue has customer){
                        dequeue business class queue
                        output starting processing info
                        calculate and store waiting time
                        signal()
                        unlock_mutex()
                        wait (processing customer)
                        output finishing processing info
                }
                else{
                        dequeue economy class queue
                        output starting processing info
                        calculate and store waiting time
                        signal()
                        unlock_mutex()
                        wait (processing customer)
                        output finishing processing info
                }
        lock_mutex()
        }
        unlock_mutex()
        pthread_exit();

}
```