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

There will be an individual thread for each customer and there will be an additional 5 threads for each clerk. Customer threads will be used to serve customers which are coming through. Clerk threads will be used to signal customer thread when to serve the customer

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

The threads work independently and there is no controller thread

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

There will be one mutex to guard the main customer queue, five mutexes to guard each clerk. For a total of 6 mutexes

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

The main thread will create all the customer and clerk threads and then will remain idle and wait for customer and clerk threads to exit.

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

Customers are entered into a FIFO queue. A new customer struct is created which holds data pertaining to each customer.

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

Any modification to the data structure will be guarded by implementing mutex lock

7. How many convars are you going to use?

There will be 2 convar for each different queue - economy and business; with an additional 5 convar for clerks

For each convar:

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

Customer convars are used to signal clerk that a customer has arrived for check in and also used to wait upon a clerk to send signal.

Clerk convars are used to signal customer that it is available to serve a customer

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

Queue mutexes are associated with queue convars and clerk mutexes are associated with clerk convars

8. Briefly sketch the overall algorithm you will use

Main thread reads the file extracts relevant details and initializes variables
After it initializes all mutexes and convars
Then it creates all customer threads and clerk threads
Waits for customer threads to terminate, then compile stats and prints

The customer threads initially sleeps to simulate arrival time
Mutex locks the queue
Puts customer into relevant queue - business or economy
Waits for signal from clerk that it is ready to start serving
When it receives signal, simulates serving time by sleeping for given serving_time
Unlocks mutex
Signals clerk that it has finished serving

The clerk thread constantly loops
It keeps checking if any queue has customers. Gives preference to business class customers
If there is a customer in queue, mutex locks the queue, signals the associated queue, unlocks the queue
Locks the clerk mutex
Waits for customer to finish check in
Unlocks the clerk mutes