1) I am going to use 2 threads, the first one is for the customers while the second thread will be for the clerks.
2) The threads work independently, there's no overall controller thread.
3) I will be using 2 mutex one for each queue and 4 mutex one for each clerk, and 2 mutex for getting time so altogether 8 mutex.
4) Yes, it will be waiting for all the customer thread to terminate.
5) Customers are represented by user id, arrival time, class type (business or economy class) and service time and I will be using a queue linked list.
6) I will be using a pthread mutex lock to prevent concurrent modification of my data structure.
7) one condition variable for each queue (2 in total) and then one condition variable for each clerk (4 in total).
    a) The condition variable for the customers will be used to alert the clerks that a customer has entered a queue and need service While the clerk condition variable will be used when clerk is done servicing a customer.
    b) conditional variables for the queue are associated with the mutex for the queue while the clerk condition variable is associated with the clerk mutex, when a customer alerts the clerk it unlocks the queue mutex in the clerk thread and vice versa.
    c) The operation to be performed when a pthread_cond_var gets a signal or is unblocked is it reenters or re-iterates in the loop and checks a condition and either skips the pthread cond_var or enters again.
8)

    Initialize global variables, pthread_mutex, pthread_condvar, attr and value.
    Read file
    Create clerk thread
    Create customer thread
    In customer thread, customer I arrives and mutex lock and customer enters queue,
    Then in a while loop waits for clerk to signal
    In clerk thread, clerk loops in and out of queue mutex lock and unlock waiting for customer signal,
    When signal is received clerk enters mutex of clerk services customer.
    Then customer thread leaves mutex and gets dequeued and moves to be serviced by clerk, clerk waits until customer finished then leaves mutex and loop.