Yilin Dong
V00928413

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

    one thread per customer and one thread per clerk. Customer threads determines when a customer enters the queue and start waiting, clerk threads
    determines when are the clerks free.

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

    they work independently

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

    i'm planning on put one for customer function and one for each clerk in the functions
    customers one would guard time and allows the clerk that's serving him changes his stat
    clerk ones would stop them when no one in queue

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

    main threads will initialize the condition variables and mutex; creat threads for clerks and customers; join them

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

```
use struct customer_inf;
int user_id;
        int class_type; //(0 = economy class, 1 = business class)
        int service_time;
        int arrival_time;
        int clerk_id; // which clerk is serving the customer
        double start_time; // start time for waiting
        double end_time; // end of customer's waitting time
        int position; // the position of the customer
```

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

    use mutex locks

7. How many convars are you going to use? For each convar:
(a) (b) (c)
Describe the condition that the convar will represent. Which mutex is associated with the convar? Why?
What operation should be performed once pthread cond wait() has been unblocked and re-acquired the mutex?

    a) when clerk selecting from the queue
    b) the mutex for the queue will be associated, make sure the clerk select the customer will not happen at the same time when customer enters the queue,
        make sure the clerk waits untill it finish serving, then go serve another guy
    c) after it's unlocked, the customer would be servised, update waittime, sleep for service time, outputs, cond_signal notice that the cler is free to
        serve another customer

8:

```
main :
    read file and save as struct customer_info
    initialize the con var and mutex
    creae threads for each clerks
    create threads for each customers
    join them
customers:
    insert to the queue
    usleep it's arrive time
    output when it arrived
    update length of the queue
    mutex lock
    output it enters the queue
    cond wait
    unlock
    find out it's clerk
    update time
    usleep for the service time
    output it's serivce ended
    cond signal
    return null
clerk:
    lock
    check the busines queue and then check the economy queue
    if queue not empty :
        broadcast
        unlock
    else :
        unlock
    wait end of serve the customer
    return NUll
```