Hatice Melike Ecevit
CMPE322
2016400138

Flight Reservation System Simulator

Project 2

In this project I used 3 mutex locks. First I will start with my design. I create the client threads and give them numbers as I create them. Then as client threads start to wake up, they take the lock on available seats where I keep all of the available seats and randomly pick one and delete it from the available seats vector. Then it creates a client_seat pair which is basically for requested seats. After that it takes the lock on requested seats and add the pair into it. It releases the lock and when it releases the server thread detects the added request via findInReservedSeats method. In that method, I look through every entry in reserved seats vector and while I'm looking, I take the look and release it. If I find the given client number, I return the index of the request and server thread prints it to output file also by taking a lock and releasing it. My design is very simple and gets the job done. The only part that I am not very comfortable with is the findInReservedSeats methods and the reason is it takes and releases a lock in every iteration and also I choose to take a lock when client picks a seat because it would be much slower towards the end if user rolls a random and then takes the lock and finds the appropriate seat.

If you are on Linux, you can compile and run it via;

```
g++ -pthreads main.cpp
```

```
./a.out -numofseats
```

You can also run it with a makefile;

```
make
./a.out -numofseats
```