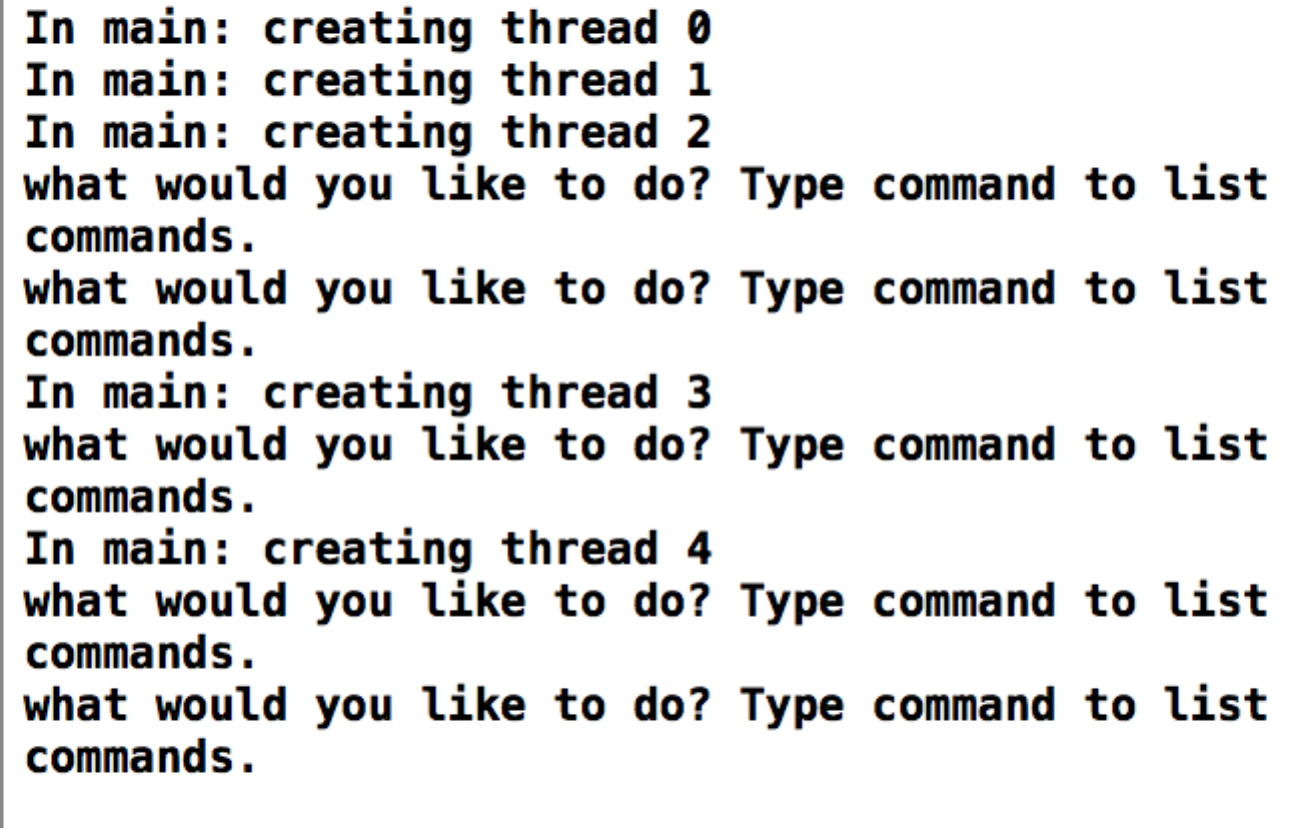
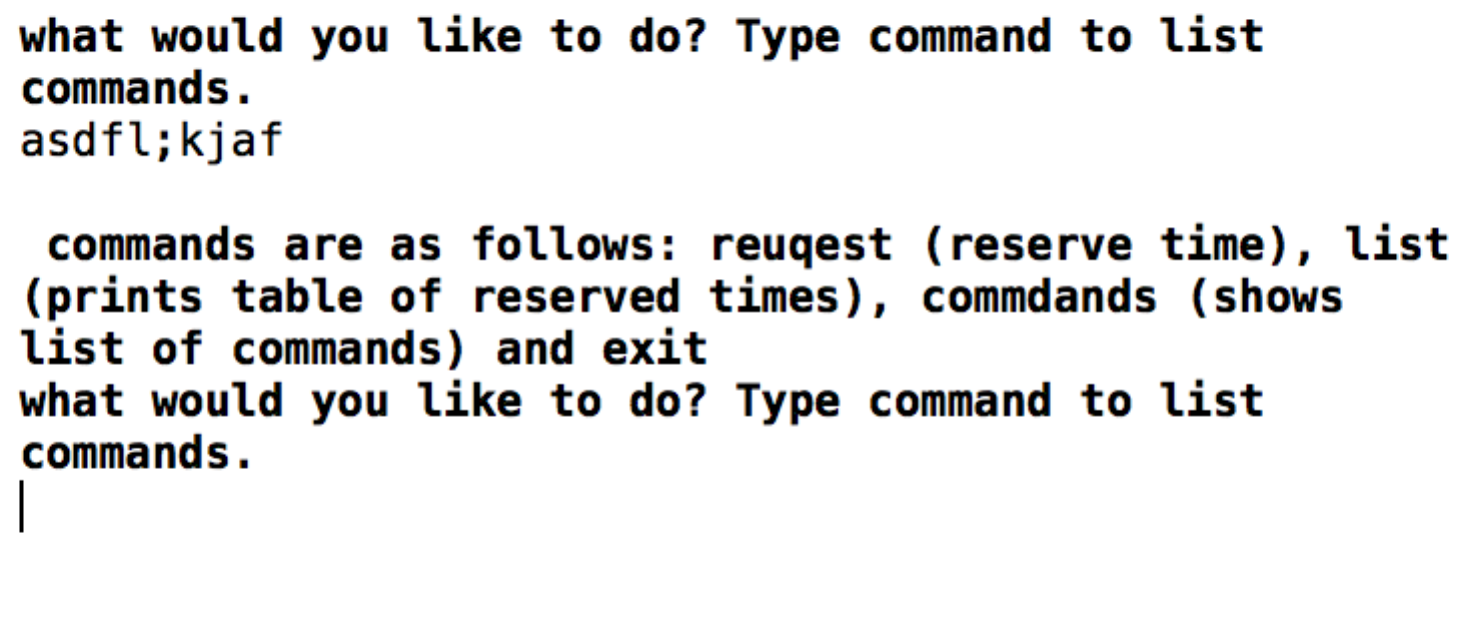
Jake Orben

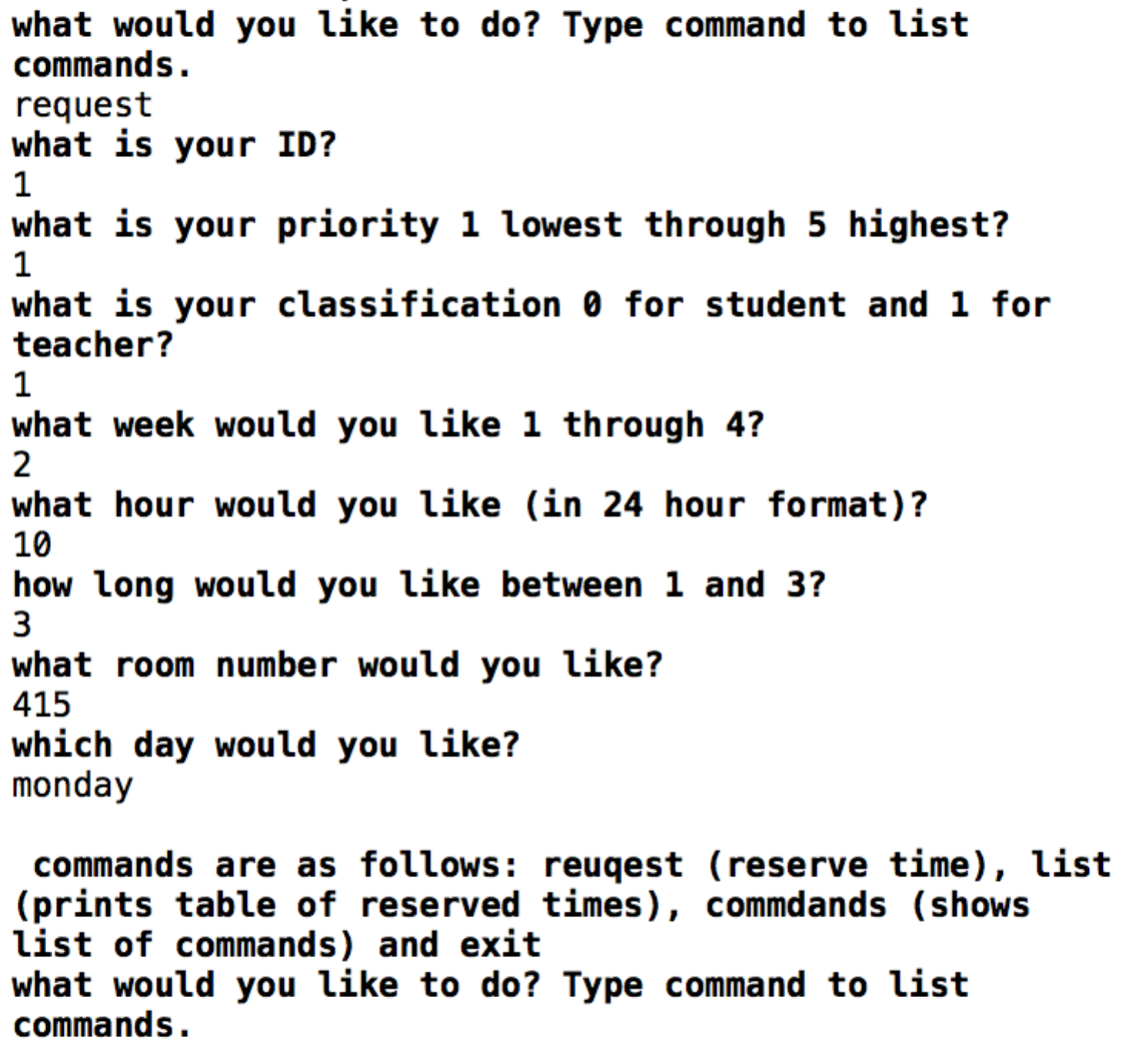
There are currently no bugs in the system. The only difficulty occurs when data occurs when multi-threading, running the threads in a for loop causes the program to print out all thread output at the same time.



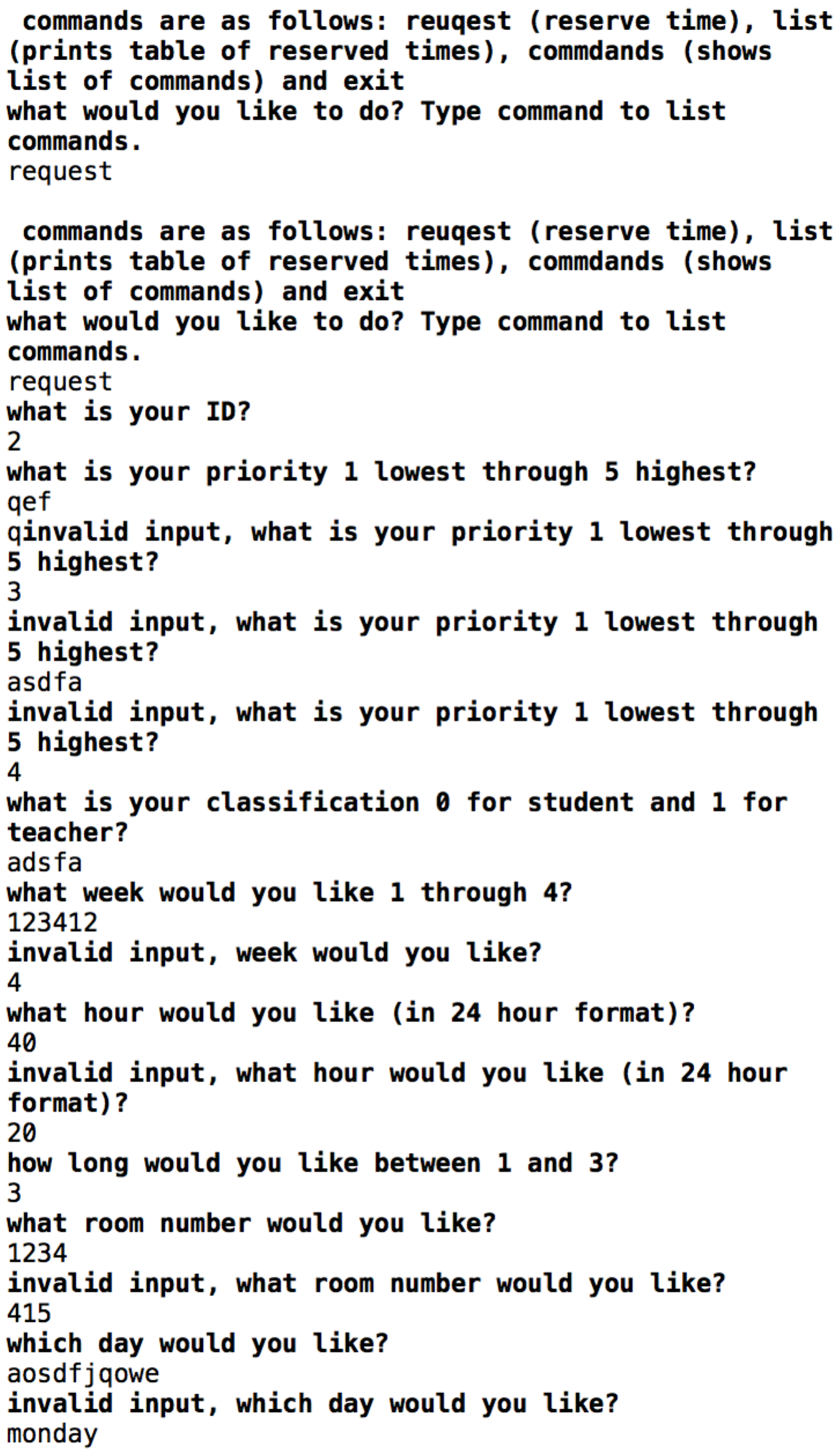
The implementation is as follows. I created a set of structs to hold information about such things as week day and reserved time. I also made a week long struct to hold all of the encompassing data from the weekdays and time structs. After, I created an array that holds all of the available rooms, these arrays are held inside of the weekday structs. After, I created a while true loop that will continually ask a given user logged into the system what they would like to do, including listing used time, creating a new time request, listing commands and exiting the program, which also exits the given thread. All data is stored in an array, which any user can access, however is deleted after the program server terminates. The array works as such, there is an array of size 4 for the weeks, inside of the weeks there are 4 arrays of length 26, the 4 arrays represent the days and the 26 represents the number of rooms, rooms hold arrays for time and if it is taken by another user as well as that user’s priority and ID. The number of threads created can be edited to fit a given situation, by adding a for loop, if necessary. There is also a considerable amount of error checking in my implementation, if a user does not respond to questions directly, they are asked the question again, until the user enters the correct data. The exception to this rule is if a given user asks for a time outside of the hours open on a given day, this will result in them being thrown back to the beginning of the while loop.



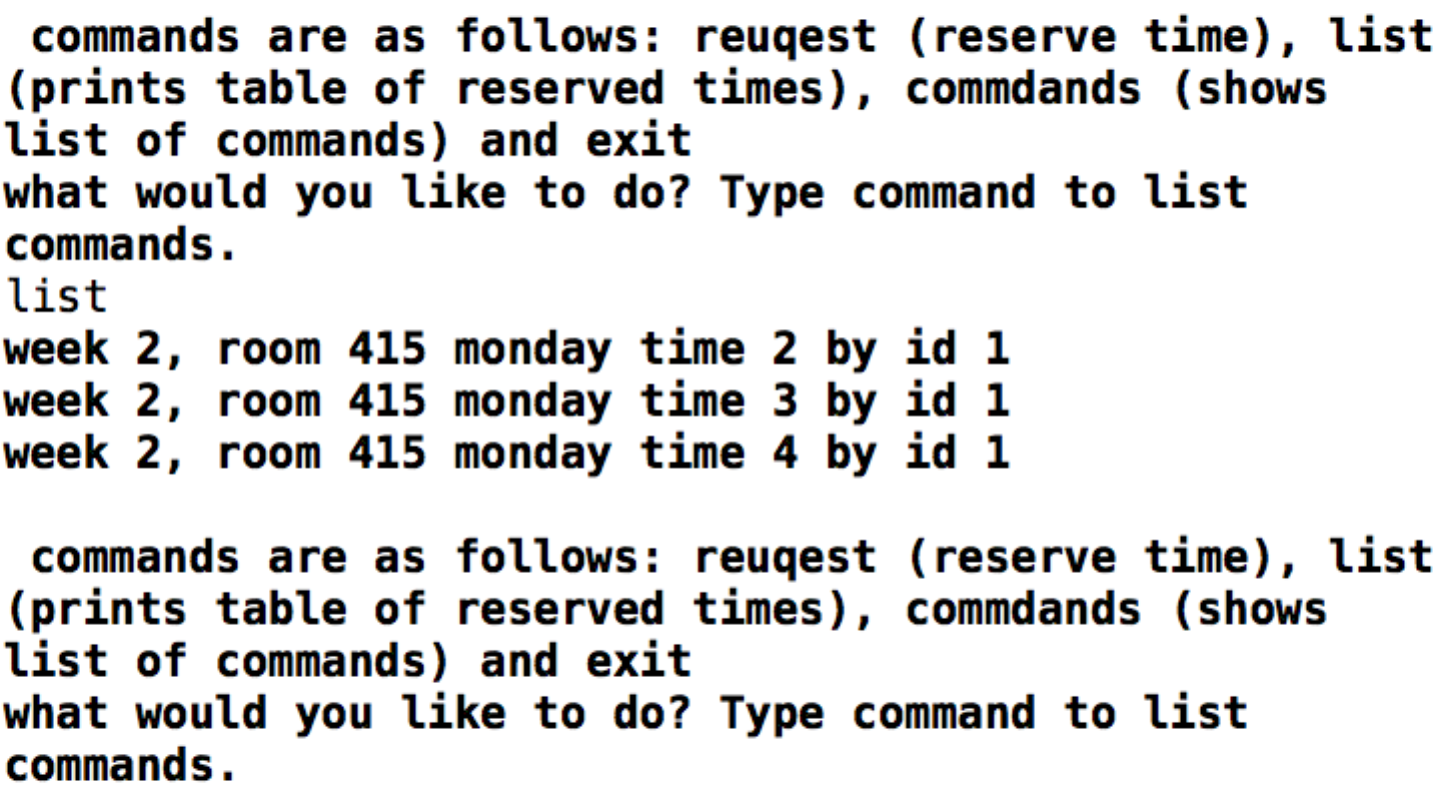
checks for invalid input from user: expected output is a list of the available commands



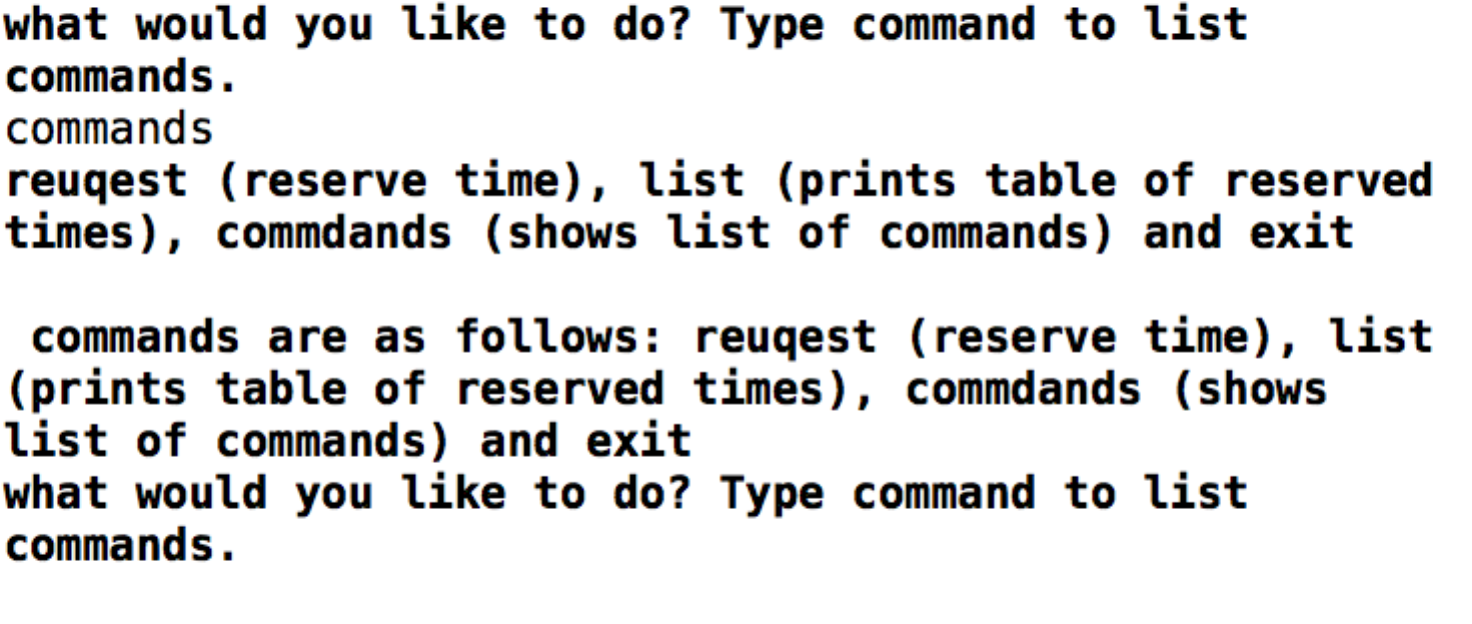
user putting in correct request for time: expected output nothing



Error catching: expected output is a repeat of the question



list: current times taken into data base: expected output is the week, room, time and ID of the data taken in by all users



asks for commands: expected output is the list of commands

Concurrency:

When using concurrent users, there can be a near infinite number of threads based on how long the for loop is. When running in concurrency mode, each request is made at the same tame, each user is given the opportunity to enter a command. After the previous user’s command is entered. All of the times from a concurrent session will be stored in the same array and any user can request to list the given times taken in the entire system. Concurrency is fully supported, however is more challenging to use, however, it still works if done properly.

Consistency:

My program is fairly consistent, the only notable issue that can occur is that, occasionally, threads are not started at the correct time, they are slightly off, meaning that sometimes that text that says creating new thread occurs after the others, in that the others enter the infinite while loop slightly sooner. This can complicate the use of the threads.