By Alexander Santander

N9684026

Statement of completeness:

I, Alexander Santander designed and developed my CAB403 Assignment entirely on my own with some minor advice from fellow peers.

[illegible]

I attempted and completed almost every task in the assignment specification.

The leader board only has the functionality required for Task 1. E.g. it only shows the statistics of the current user.

Description of the data structure that is used for the Leader Board:

A linked list of structs are used to hold the information about the client such as their request number and socket. In this same struct the number of games won and played for each user has been stored. Unfortunately due to technical difficulties the leader board only accesses the struct for the current client/user rather than iterating through all of them.

Description of how the critical-section problem is handled in Task 2

N/A

Description of how the thread pool is created and managed in Task 3

When the program starts 10 threads are created to allow for up to 10 users. These threads all call upon “handle_requests_loop”. This function essentially makes these threads wait until a request is added to the queue. Back in the “main” the server then waits for a connection and once one is made it calls upon “add_request” passing in the new socket that the client will use and the requests number which is iterated each time a new connection is made. “add_request” initializes a new struct of type “request” and assigns all relevant values, including a pointer to the next struct the structs will be linked. Now that there is a request a thread is released in the “handle_requests_loop” to then handle the request and run the game.

Instructions on how to compile and run your program

Both the .c and the executable files which have been compiled are in the submitted zip file. This will explain how to run the program including compilation (which may not be necessary).

1. Compile Server.c with the command “gcc -o Server Server.c -lpthread”
2. Compile Client.c with the command “gcc-o Client Client.c”
3. Start the server with the command “./Server #{portnumber}
4. Connect the client to the server with the command “./Client localhost #{portnumber}”
5. Repeat step 4 up to 10 times for multiple concurrent connections