**Statement of Completeness**

There was 50/50 work displayed on the assignment, it’s a bit rough to estimate exactly, since there were areas that the other person needed help with. But overall the amount of work spent for the assignment was distributed evenly.

**Task 1:** Description of the Data structure used for the Leader Board

Data structure used is an array of structure for the Leaderboard. In the Server there’s a structure that holds the *username*, *password*, *games played*, *games won*. In the client there’s a structure like the server, but without the password. So, when the client chooses to see the leaderboard, the server sends the details of each person that’s played a game, the client then catches it, and performs the comparing of the names to display the leaderboard based on the specification.

**Task2:** Description of how the critical-section is handled

In the program, it’s basically the same as practical 5 of the unit, it copies exactly how the thread is created, and how the mutex locks are implemented. |||||Since the program is created instantly, right after a user finishes playing a Hangman game, it automatically writes to the struct, this is very instant, as well as the viewing of the leaderboard, so when the user requests the leaderboard, it instantly sends the users that have played the game.

**Task3:** Description of how the thread pool is created and managed

Same as task 2, thread pool is created similarly to practical 5 of the unit, it creates a thread of 10, which handles the connections concurrently.

**How to compile the program:**

* Extract the files, open a terminal window and type ‘make’.
* The server can be started using *‘./server port#’*. Any port number can be used, port defaults to 12345 if no port number is specified.
* The client can be started using *‘./client serverIP serverPort#’*