# Process Management and Distributed Computing

## Members

Jose Legaspi – n9170081

Matthew Lye – n9191453

## Statement of Completeness

Task 1, 2 and 3 were completed.

## Statement of Contribution

Jose developed multithreading, thread pools, login, and game logic. Matt developed reading from text file, leaderboard code, and game logic.

## How to Run and Compile

Both the C programs files for the assignment were compiled using the Linux Ubuntu terminal. The line to compile the server is “gcc -o server server.c -lpthread” and “gcc -o client client.c” for the client. In order to run the server a port has to be used the line would be “./server 1234”. If no port has been specified, the server will run on port 12345. In order to connect to the server, you must pass both the server IP address and the port number as arguments. Example: “./client localhost 12345”.

## Leader Board Data Structure

The leader board data is stored in a struct named scoreBoard, which holds the players name, games won and played. Once a user had successfully logged in a struct would be initialised based on the line the user’s name is on in the authentication.txt. The player’s name was then assigned to a certain struct and each game played, depending on if it was won or not either gamesPlayed and gamesWon would be called and added to or just gamesPlayed.

## Thread Pool Creation and Management

A global array of threads is created on line 31 with a length equal to a predefined amount denoted in constant ‘MAXTHREADS’. In our main function, our ‘accept connection’ function is nested within a for loop, so that we can reference the thread pool with iterable variable ‘i’. Our threads are used when the server accepts a connection. Using pthread\_create, we pass an individual thread (from the thread pool), pass in our thread function ‘run’ (our main application function that takes us to the login section), and pass in our struct array ‘thread\_args’, which we use to store variables unique to each individual thread.

When a user quits the program, the thread is then deleted using ‘pthreads\_cancel’.