Project 1

In this project, you will write a client-server program using TCP. The client will send requests to the server, and the server will send the reply after processing the requests. The client can send the following requests to manage a simple student database stored on the server. To keep it simple, you can use a text file for the database. You can randomly use a 6-digit numerical number as a student's ID. We assume students' first and last names are less than 10 characters long. Students' information includes their ID, first and last names, and their scores (0-100) for the network course.

There are two programs here: the client program and the server program. Put the client program on **zeus** and the server program on **eros**. On the client side, you can display a menu with the following list for the client to choose a command. For example, if the client chooses '1', you should ask the client to enter ID, Fname, Lname, and score, and then send these to the server. On the server side, when it receives the related information from the client, it should add a new entry in the database. If the client does not choose '6', the menu should be redisplayed after each command is completed.

1. add(ID, Fname, Lname, score): this request adds new student information to the database.

2. display(ID): this request sends the ID of a student to the server, and the server returns the student's information to the client.

3. display(score): this request sends a score to the server, and the server returns the information of all students whose scores are above the sent score to the client.

4. display_all: this request displays the information of all the students currently in the database.

5. delete(ID): this request deletes the student entry with that ID from the database.

6. exit: this request terminates the program.

****

See the sample code for your reference to set up the TCP connection.

You do not have to adhere to the sample to make connections. You can use any that establishes a TCP connection. You should begin the program by establishing the connection first, and then determine if two machines can communicate with each other using simple messages, and then progress to more complex messages. You have flexibility in designing user interfaces as long as you meet the above basic requirements. You can write the program in C, Java, or Python.

****

Submission:

To avoid losing any files, it is best to zip them into a .zip file. Submit your project to CANVAS and include a readme text file that explains to the grader how to run your programs. Without this file, it is highly likely that, your project will not be run properly.

Please submit your program to CANVAS before the deadline. Email submissions WILL NOT be accepted.