CN Lab Assignment 4

Max Marks:20 **Deadline : 25-04-2023**

Instructions:

● **Please submit original work. Plagiarism/copying of any form, will generate zero mark. Plagiarism from the internet also should less than 8%.**

● Submit all documents in google classroom.

● Save all submission files into a single folder and submit the compressed folder.

● **Explain the working of functions in proper comments.**

● **All tasks required proper executable codes, screenshots for otput etc.**

● **No compile time and run time errors are entertained.**

● **There is no partial marking for all the given tasks.**

● **Create a readme file explaining how to execute your code.**

Must follow the following naming convention:

○ Name the ZIP file submission as,

Yourname_Rollnumber_Assignmentnumber.zip

Q1: You are asked to implement a file transfer program in Python using sockets. The client should be able to send a file to the server, and the server should be able to receive the file and save it to disk. Your task is to implement the client code that sends a file to the server. Your implementation should create a TCP socket and connect to the server at the given address and port. Then it should send the filename to the server, followed by the contents of the file. Finally, it should close the connection and return nothing. [5 Marks]

Q2: Implement a client-server chat application that allows multiple clients to communicate with each other via a central server. The chat application should support the following features: [10 Marks]

1. **User Registration: The users should be able to register themselves by providing a unique username and password. The registration information should be stored on the server.**

2. **User Login: The users should be able to log in to the system by providing their username and password. Upon successful login, the user should be able to see a list of active users.**

3. Chat Rooms: The users should be able to create, join, and leave chat rooms. Each chat room should have a unique name and a list of users who are currently in the room.

4. Messaging: The users should be able to send and receive messages in real-time within the chat rooms they have joined. The messages should be displayed in the chat room along with the username of the sender and the timestamp of the message.

5. Logout: The users should be able to log out of the system. Upon logout, the user should be removed from all chat rooms they have joined.

Write a report documenting the design and implementation of the chat application. The report should include screenshots, code snippets, and explanations of the code. Your implementation should be original and your own work. You are free to use any Python libraries or frameworks to implement the chat application. Submit the Chat application and report in a single zipped file.

Q3. Write two separate C programs, one for TCP server (handles requests for multiple users) and another one for clients. [5 Marks]

Your server program will be a multi-process server that will “fork” a process for every new client it receives. Multiple clients should be able to simultaneously chat with the server and the connection should be persistent. You may use multithreading using pthread or fork() system call to implement the server.

The protocol between the client and server is as follows:

1. The client connects to the server, and then asks the user for input. The user enters a simple arithmetic expression string in postfix form (e.g., "1 2 +", "5 6 22.3 * +"). The user's input is sent to the server via the connected socket.
2. The server reads the user's input from the client socket, evaluates the postfix expression, and sends the result back to the client as well as writes the following in a file named “server_records.txt”. [at the beginning create an empty file]
<client_id> <query> <answer> <time_elapsed>
Make sure to handle the race conditions while writing to this file at the server.
3. The client should display the server's reply to the user, and prompt the user for the next input, until the user terminates the client program.
Server should be able to handle addition, multiplication, subtraction, and division operations

(postfix form is space separated).

Sample test cases are:

User types: 1 2 +, server replies 3

User types: 2 3 *, server replies 6

User types: 4 7 3 + -, server replies -6

User types: 30 1.0 /, server replies 30.0

Below is a sample run of the client.

```
$ gcc client.c -o client
$ ./client
Connected to server
Please enter the message to the server: 22 44 +
Server replied: 66
Please enter the message to the server: 3 4 *
Server replied: 12
```

Sample server_records.txt :

```
1 1 2 + 3 4
1 3 4 - -1 7
2 2 3 / 0 9
3 4 4 / 1 11
1 3 4 * 12 20
2 8 9 + 17 21
1 1 2.0 / 0.5 23
```