Assignment Report

Language: Python 3.7 (Assignment Completed Through SSH to CSE servers)

Program Design:

Data Structure used:

- Lists
  - Used to store all the users connected to the server
  - Used to store details of threads that were created on the server
- Dictionaries
  - Used to store references to threads that were used for multiple concurrent users. (multithreading)

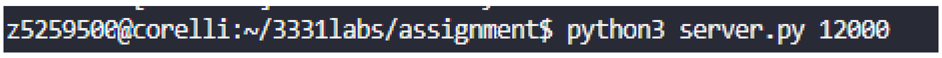
Features/Operations Implemented:

- Implemented all commands:
  - CRT: Create Thread
  - LST: List Threads
  - MSG: Post Message
  - DLT: Delete Message
  - RDT: Read Thread
  - EDT: Edit Message
  - UPD: Upload File
  - DWN: Download File
  - RMV: Remove Thread
  - XIT: Exit

How to start each program:

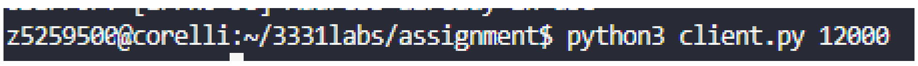
PORTS NEED TO BE THE SAME

Server.py Usage: python3 server.py <port>

E.g., 

```
z5259500@corelli:~/3331labs/assignment$ python3 server.py 12000
```

Client.py Usage: python3 client.py <port>

E.g. 

```
z5259500@corelli:~/3331labs/assignment$ python3 client.py 12000
```

How Program Works:

Upon running the client program when the server is up, the client is asked to enter in a username to login with. The username entered is sent to the server to satisfy a set of conditions before it provides a response for the user to enter in another username or to enter the password.

Upon successful authentication, the user is presented with a list of various commands that they can use to interact with the forum application.

Each input command is checked before sending it to the server, if it fails the conditions set, nothing would be set, and the user will be prompted to enter the same/different command.

Multiple users can connect to the server at the same time efficiently using multithreading. Messages are processed in the order of what they arrive at the server (first in first serve.)

Upon exiting as a client via the command 'XIT', the client's program would send a data segment to the server to remove the client from the threading and active users list before the clients program closes itself.

Application Layer Message Format:

In the current design, messages are communicated as a string format between the server and clients. Often, messages would have the command/forum operation at the beginning of the string for the string to be easily split and identified as to what command was sent. Cases where commands weren't sent, conditional statements were used to differentiate different possible responses.

A design consideration for the future to increase the reliability of UDP communications through the application layer is to use a JSON format where extra information could be included such as SYN/ACKs, separation of commands from data body and to use timeouts for any lost packets.

Functionality to handle concurrent users:

For handling multiple concurrent users, I used the threading library which allows for separate flows of execution (executing command in parallel). Upon the connection of a new client, a new thread is created to process the commands sent by each respective client. The references to each thread was stored in a dictionary as key value pairs where the key was the clients connection address and the value being the thread.

When the server socket receives data from different clients, it finds the correct thread to send the data to for the data to be processed.

Problems with programs:

- If a client uses 'ctrl + c' to stop their program, the server will never know that the client is disconnected. The associated thread that processes all the commands for that client will remain open. Server may experience undefined behaviors.
- Reliability for UDP wasn't implemented therefore any lost packets may likely prevent the server and client from continuing to run.

Input Assumptions:

- Users are expected to only try and upload a file that they had on their computer.

- All arguments can contain upper- and lower-case characters, numbers and ‘.’ (as is listed in the specification)

References:

1. The sample code provided by the tutor Wei Song.
    a. https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70107
    b. https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70108