Assignment - 2
Blocking vs Non-blocking Socket

Problem Description

Problem 1-

Write two separate C programs, one for **TCP server** (handles requests for multiple users) and other one for client.

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.

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>
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.

At the very minimum, server should be able to handle addition, multiplication, subtraction, and division operations (postfix form is space separated).

Problem 2-

Write two separate C programs, one for **TCP server** (handles requests for multiple users) and other one for client.

Your server program will be a single process server that uses the **"select**" system call to handle multiple clients. Multiple clients should be able to simultaneously chat with the server.

The protocol between the client and server is as follows:

4. 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.
5. 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>
6. 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.

You may also assume that the postfix expression is separated by a space. So, some 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
...
```

...

In parallel, here is how the output at the server looks like this (you may choose to print more or less debug information).

```
$ gcc server.c -o server
$ ./server
Connected with client socket number 4
Client socket 4 sent message: 22 44 +
Sending reply: 66
Client socket 4 sent message: 3 4 *
Sending reply: 12
...
...
```

Marking Scheme

Total - **100 Marks**

Problem 1-

Handling Concurrency - **15 Marks**
Marks For handling postfix queries - **20 Marks**
Error handling strategies- **10 Marks**

Problem 2-

Handling Concurrency - **15 Marks**
Marks For handling postfix queries - **20 Marks**
Error handling strategies- **10 Marks**

Coding style - 10 Marks.

Submission Deadline: January 14, 2020 [2 PM]

Submission Guidelines

1. This is an individual assignment.
2. Create a folder where the folder name is “<roll_number>_Assgn2”.
3. Create two subfolders under the folder - problem1 and problem2.
4. Compress the parent folder (whose name is your roll number), upload compressed folder at Moodle.

Reference Links

1- Blocking vs Non-blocking call
https://www.scottklement.com/rpg/socktut/nonblocking.html

2- Select system call
http://www.tutorialspoint.com/unix_system_calls/_newselect.htm

3- Beej Guide
https://beej.us/guide/bgnet/pdf/bgnet_usl_c_1.pdf

HAPPY CODING!! :)