NES416 - Network Programming
Programming Assignment 1

Objectives:

- To write a complete TCP client-server application.

Problem Statement:

In this assignment, you will use the socket API to write both a TCP client and a concurrent TCP server programs. The requirements of this distributed application are as follows:

- The server accepts a single command line argument: the port number it listens and accepts connections on, e.g., 5001.

- The server performs the following five operations on integer-based arguments passed to the server by clients, and the server returns the result to the calling client consequently:

    1. Computing the count of numbers that consist of 1 digit in a list of integers.
    2. Computing the count of numbers that consist of 2 digit in a list of integers.
    3. Finding the maximum number in a list of integers.
    4. Finding the maximum number in a list of integers.
    5. Finding the difference between the maximum and the minimum numbers in a list of integers.

- In case of unsupported operation, the server replies to the client with an error message.

- The server keeps an access log-file for every request, in which it records the associated server's local date and time, the client's IP address, the requested operation, and whether the operation was carried out successfully or not.

- The server can serve up to a maximum of 5 clients concurrently. Other clients should be kept waiting until one or more clients being served finish.

- The client accepts four command line arguments:

    – The server IP address, e.g., 127.0.0.1,
    – The server port number, e.g., 5001,
    – An integer representing the number of micro-seconds the client waits between sending subsequent requests to the server, e.g., 200000, and
    – A name of file to read the input from, e.g., ./input.txt. The format of the file is defined below.

- After initiating a TCP connection to the server, the client

  1. Reads one line from the input file,
  2. Show the available operations on standard output and Read the selected operation from user.
  3. Sends the request to the server,
  4. Prints the result on the standard output, e.g.,
  OP1(12, 33, 9, 5, 2, 53) = 3
  OP2(12, 33, 9, 5, 2, 53) = 3
  OP3(12, 33, 9, 5, 2, 53) = 53
  Unsupported Operation
  OP4(12, 33, 9, 5, 2, 53) = 2
  OP5(12, 33, 9, 5, 2, 53) = 51
  5. Waits (sleeps) for the given number of micro-seconds (per the command line argument), and
  5. Repeats steps 1–5 until the end of file.

- Each line of the input file contains 6 space-separated integers, each integer is one or two digits. e.g.

  12, 33, 9, 5, 2, 53
  6, 3, 59, 15, 82, 45

- If the user selects an operation that is not exist, the server returns an error message to the client, which in its turn prints this on the standard output. For example, the user select 10 or -3, the server must return an error message, e.g., "Unsupported Operation" to the client
- The client sends multiple requests (i.e., corresponding to multiple lines in the file), and receives the associated results on a single TCP connection with the server.

Hints:

- You are not allowed to use libraries or tools developed by any third party.
- Put your source files only (no binaries) along with a plain-text file explaining how to compile and run your code in a single compressed file.
- Submit your compressed file using the link of the E-learning site.
- Zero grade will be assigned for all programs that will not be compiled or run correctly.
- Partial credit is given only for working code that does not implement all the requirements.
- **One submission is needed per group.**
- **Comment your code as needed.**