IIT Goa CS 378 (Computer Networks LAB)
Instructor: Neha Karanjkar

LAB 2: Socket Programming Using Python

Instructions:

- For this lab you can work individually OR in teams of 2 persons. Choose your teammate. If working in a team, both team members need to understand the approach and the solution.
- Both persons in the team should work on each problem together and in sequence. **Do not** divide up the questions among team members.
- You **are allowed** to use Google search and the www for information.
- You **can** help other teams. However, remember that there is a time limit. Complete your own work first before helping others. Help others by sharing the method and the knowledge to arrive at a solution.
- **Do not share your code with another team.** If you copy another team's code OR allow your code to be copied by another team you will get 0 marks for this exercise and a double grade penalty for the course in addition.
- There is a strict time limit of 2 hours. You will be graded for the exercises you can complete within this time budget.
- When you complete all the exercises (or as many as you can within the time budget) please notify the instructor for a demo of the solutions.

ROLL NUMBERS (Do not write names):

_____(Team member 1)

_____(Team member 2, if any)

1. Write Client and Server programs using Python that communicate over sockets and perform the tasks described below. (Figure out what type of sockets you need to use.)

   a) Choose some unique name for your Client process. The Client first establishes a connection with the Server. After this, it sends the server its name. Upon receiving the name, the Server should print "Connection initiated by <client name>" and send an "ack" string to the client.

   b) After establishing the connection and getting the "ack" from the Server, the Client process runs an infinite loop in which it does the following:
      1. Prompts the user for a command (a line of text entered through the keyboard).
      2. Acts on the command as follows:
         - If the command is "q" (short for quit), the client program closes the connection and exits.
         - If the command is "a" (short for add), the client program chooses two numbers (each between 1 to 10) at random and sends these two numbers to the server, **bundled up into a single string**. It also prints the two numbers on the screen.

- The Server, upon receiving this message, separates out the two numbers, adds them up and sends the answer back to the client.
  The Client prints “answer received from Server = <answer>” upon receiving this response.

- If the user enters any command other than “q” or “a”, the Client prints “Invalid command” and prompts the user again.

2. For the same Client program as in question 1, modify only the Server program so that a single Server process is able to **communicate with multiple client processes at a time**. (Remember, each client process should have a unique name even though they are all instances of the **same** program. To do this, you may append some randomly chosen number to the name.) For each connection, the Server should provide the same service as described in question 1.

   Hint: There are several ways to implement this. One way is to use mutiple **threads**, one per connection. A skeleton for a multi-threaded server program can be found in this answer on StackOverflow: https://stackoverflow.com/a/40351010/1329325

3. Start up Wireshark and apply a filter such that only the traffic generated by your Client and Server processes is displayed. Identify the messages used during the Handshake and the actual text sent by the two processes. Show a demo to the instructor.