Programming Assignment 1: Socket Programming

In this assignment, you'll write a client that will use sockets to communicate with a server that you will also write. Here's what your client and server should do:

Your client should first accept an integer between 1 and 100 from the keyboard, open a TCP socket to your server and send a message containing (i) a string containing your name (e.g., "Client of John Q. Smith") and (ii) the entered integer value and then wait for a sever reply.

Your server will create a string containing its name (e.g., "Server of John Q. Smith") and then begin accepting connections from clients. On receipt of a client message, your server should

i. print (display) the client's name (extracted from the received message) and the server's name
ii. itself randomly pick an integer between 1 and 100 and display the client's number, its number, and the sum of those numbers
iii. send its name string and the server-chosen integer value back to the client

Your client should read the message sent by the server and display its name, the server's name, its integer value, and the server's integer value, and then compute and the sum. The client then terminates after releasing any created sockets. As an aside (and as a check that you are doing things correctly, you should make sure for yourself that the values and the sums are correct!)

What to turn in:
Upload a zipped folder to Canvas that contains your client and server scripts, and a README file. Guide to Creating a README File Your code should be appropriately commented. You should program your client and server to each print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc.), so that you can see that your processes are working correctly (or not!). This also allows me to quickly determine from this output if your processes are working correctly.

Programming notes

Here are a few tips/thoughts to help you with the assignment:

- You must chose a server port number greater than 1023 (to be safe, choose a server port number larger than 5000). If you want to explicitly choose you client-side port, also choose a number larger than 5000.
- Make sure you close every socket that you use in your program. If you abort your program, the socket may still hang around and the next time you try and bind a new socket to the port ID you previously used (but never closed), you may get an error.