Name - Naishil Shah
Cruzid - nanshah@ucsc.edu

Testing:

To test the application:
(Assuming you are in the parent directory which contains all the above listed directories)

1. Running the Python Test Script - Enter the following commands in the terminal
    a. cd src
    b. python3 test.py

The output should say all the tests ran successfully.

2. Providing individual commands i.e. running the server and client application independently.
    a. Open Terminal 1
        i. cd bin
        ii. ./myserver <portnumber>
    b. This will start the server
    c. Open Terminal 2
        i. cd bin
        ii. ./myclient 127.0.0.1 <portnumber>
    d. This should establish a connection between the server and the client.
    e. Enter test commands on the client terminal on the probe of '< '
    f. Enter 'exit' to terminate the connection from the client side.

Workflow Description of the code:

1. In the main function of both the server and client respectively, I invoke 2 other functions.
2. Both the main functions in the server and the client first call the estab_connection.
3. Once the connection is established, the main function then calls the transfer_data function which is responsible for exchanging commands and their respective outputs between client and the server.
4. The connection is not terminated until the exit command is entered.
5. This implies that multiple data commands are transferring the data via the same connection.
6. As soon as the exit command is encountered, the client connection is terminated and the server is ready (open) for a new connection.

Name - Naishil Shah
Cruzid - nanshah@ucsc.edu

Assumptions:

The following assumptions are made :

1. Both the command and the server's output character limitation is set to 500.
2. The connection between the client and the server is established only once and not repeatedly. The single connection is used to make all the data transfer between them.
3. The error regarding various errors and client/server misbehaviours, are handled by the application by printing out appropriate error messages for the user on the terminal screen.
4. In case of bad commands or invalid data entries, the server does not respond to the client and hence the user does not get any output.
5. The server created here is an iterative server which only serves a single client at any given time.

General Comments:

1. The type of sockets used in the client and server application are called Stream Sockets. They guarantee in order transmission of data at the receiving end, i.e. Transmission Control Protocol (TCP) is used here.

2. The handshaking process happening between server and client is as follows:
    - Both the server and the client have to first create a socket. They use the ***socket()*** system call for this.
    - The server has to then perform 2 additional steps before it can start to accept any client. The first step is to call the ***bind()*** function. This assigns a local protocol address to the socket. The second step is to invoke the ***listen()*** function which converts an unconnected socket into a passive state socket. This indicates the kernel that it should start accepting incoming requests and also specifies the maximum number of connections which are acceptable by the kernel.
    - The client uses ***connect()*** function to connect to the server. On the server side, the server then accepts the connection from the client using the ***accept()*** system call.
    - This completes the process of establishing a connection between the client and the server. Both of them can now use ***read()*** and ***write()*** system calls to transfer data between them.
    - The application finally terminates the connection between the client and server by using the ***close()*** system call.