Computer Networks
Assignment 6

Name: **Krunal Rank**
Roll No: **U18CO081**

Create a chat application involving one server and multiple clients.
Part 1:
1) Use Socket Programming and TCP Protocol.
2) Configure your server in such a way that it can be connected to N different clients; one of which will be chosen as the “primary client”.
3) Any client who wishes to communicate with the server will send message to the primary client; who will then forward it to the server. (Clients other than primary one can only receive messages from the server, but cannot send any.)
Part 2:
4) Store multiple programs on the server; involving various string operations.
5) Upon any connection, the server will present the entire list of functions to the client.
6) Client should be able to choose one function from the above list and accordingly the server will take input arguments from the client and display relevant output.
(Note: Here all the clients can communicate with the server directly.)

Part 1:
Server:

```
'''
   Author : KRHero
   IDE : VSCode
'''
import socket, threading
class ClientThread(threading.Thread):
   def __init__(self,clientAddress,clientsocket,id):
       threading.Thread.__init__(self)
       self.csocket = clientsocket
       self.id = id
       self.clientAddress = clientAddress
       print ("New connection added: ", self.clientAddress)

   def run(self):
       print ("Connection from : ", self.clientAddress)
```

```
        data = self.csocket.recv(2048)
        response = data.decode()
        if response=="Client Initial Handshake":
            print ("Initial Handshake successful from Client: ",self.clientAddress)
            self.csocket.send(bytes("Server - Connection established!" ,'UTF-8'))
        else:
            print ("Initial Handshake failed from Client: ",self.clientAddress)
            return

        while True:
            data = self.csocket.recv(2048)
            response = data.decode()
            response = response.strip()
            print("Response: ",response.strip())
            if response == "exit":
                self.csocket.send(bytes("Bye!",'UTF-8'))
                break

            self.csocket.send(bytes("Server - "+response ,'UTF-8'))


        print ("Client disconnected: ",self.clientAddress)

LOCALHOST = "127.0.0.1"
PORT = 8080
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((LOCALHOST, PORT))
print("Server started")
print("Waiting for client request..")
sockets = dict()

while True:
   server.listen(1)
   clientsock, clientAddress = server.accept()
   newthread = ClientThread(clientAddress, clientsock,0)
   newthread.start()
```

Client:

```
'''
   Author : KRHero
   IDE : VSCode
'''

import socket
SERVER = "127.0.0.1"
PORT = 8080
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((SERVER, PORT))
client.sendall(bytes("Client Initial Handshake",'UTF-8'))
while True:
   in_data =  client.recv(1024)
   print("Response: " ,in_data.decode())
   if in_data.decode()=="Bye!":
       print("Connection disconnected!")
       break
   out_data = input()
   client.sendall(bytes(out_data,'UTF-8'))
client.close()
```

Output:

Part 2:

Server:

```
'''
   Author : KRHero
   IDE : VSCode
'''



import socket, threading
class ClientThread(threading.Thread):
   def __init__(self,clientAddress,clientsocket):
       threading.Thread.__init__(self)
       self.csocket = clientsocket
       self.clientAddress = clientAddress
       print ("New connection added: ", self.clientAddress)
   def run(self):
       print ("Connection from : ", self.clientAddress)
       msg =   '''
String Operations

Commands
upper <string> : Converts <string> to uppercase
lower <string> : Converts <string> to lowercase
find <string1> <string2> : Finds <string2> in <string1>
length <string1> : Returns length of <string1>
concat <string1> <string2> .... : Concats all the given strings as args

Help
help : Displays this command menu

Exit
exit : Close current socket connection

               '''


       self.csocket.send(bytes(msg,'UTF-8'))
       data = self.csocket.recv(2048)
       response = data.decode()
       if response=="Client Initial Handshake":
           print ("Initial Handshake successful from Client: ",self.clientAddress)
       else:
           return
```

```
    while True:
        try:
            data = self.csocket.recv(2048)
        except:
            break
        response = data.decode()
        params = response.split(' ')
        print("Response from ",self.clientAddress,": ",response)
        if params[0]=='upper':
            if(len(params)<2):
                self.csocket.send(bytes("Insufficient args!",'UTF-8'))
                continue
            for i in range(2,len(params)):
                params[1]  = params[1] + ' ' + params[i]

            self.csocket.send(bytes(params[1].upper(),'UTF-8'))
        elif params[0]=='lower':
            if(len(params)<2):
                self.csocket.send(bytes("Insufficient args!",'UTF-8'))
                continue
            for i in range(2,len(params)):
                params[1]  = params[1] + ' ' + params[i]
            self.csocket.send(bytes(params[1].lower(),'UTF-8'))
        elif params[0]=='find':
            if(len(params)<3):
                self.csocket.send(bytes("Insufficient args!",'UTF-8'))
                continue
            val = params[1].find(params[2])
            self.csocket.send(bytes(str(val),'UTF-8'))
        elif params[0]=='length':
            if(len(params)<2):
                self.csocket.send(bytes("Insufficient args!",'UTF-8'))
                continue
            for i in range(2,len(params)):
                params[1]  = params[1] + ' ' + params[i]
            self.csocket.send(bytes(str(len(params[1])),'UTF-8'))
        elif params[0]=='concat':
            if(len(params)<3):
                self.csocket.send(bytes("Insufficient args!",'UTF-8'))
                continue
            res = ""
```

```
                for i in range(1,len(params)):
                    res = res + " " + params[i]
                self.csocket.send(bytes(res,'UTF-8'))
            elif params[0]=='help':
                self.csocket.send(bytes(msg,'UTF-8'))
            elif params[0]=='exit':
                self.csocket.send(bytes("Bye!",'UTF-8'))
                break
            else:
                self.csocket.send(bytes("Invalid Command!",'UTF-8'))


        print ("Client disconnected: ",self.clientAddress)

LOCALHOST = "127.0.0.1"
PORT = 8080
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((LOCALHOST, PORT))
print("Server started")
print("Waiting for client request..")
while True:
    server.listen(1)
    clientsock, clientAddress = server.accept()
    newthread = ClientThread(clientAddress, clientsock)
    newthread.start()
```

Client:

```
'''
   Author : KRHero
   IDE : VSCode
'''


import socket
SERVER = "127.0.0.1"
PORT = 8080
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((SERVER, PORT))
client.sendall(bytes("Client Initial Handshake",'UTF-8'))
while True:
   in_data =  client.recv(1024)
   print("Response: " ,in_data.decode())
   if in_data.decode()=="Bye!":
       print("Connection disconnected!")
       break
   out_data = input()
   client.sendall(bytes(out_data,'UTF-8'))
client.close()
```

Output: