Assignment 4

Implementation of TCP Socket Programming

NAME: Shirish Manoj Bobde

Reg. No.: 812

Roll No.: ECE/21152

Problem Statement 1

Write a TCP socket program (in C/C++/Java/Python) to establish connection between client and server. The client program will send an input value n to the server and the server program will return the sum of the square of first n natural numbers. Client will display the value send by server. The communication between client and server will continue until client send 'Quit' message to the server.

Code

Server

```
import socket
port=50000
host="127.0.0.1"

server= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((host, port))
print("socket binded to %s" %(port))

server.listen(2)
print("Socket is listening...")

# Accepting/Establishing connection from client.
conn, addr = server.accept()
print('Got connection from', addr)

while True:
    recieved_data = conn.recv(2048)
    #print("Message from client: ",recieved_data.decode())
    if recieved_data.decode()=='quit':
        break
    else:
        n = recieved_data.decode()
        n = int(n)
        i=1
        sum = 0
        for i in range(n+1):
            sum += i*i
```

```
            print(i)
        print("Sum: ",sum)

        sum = str(sum)
        conn.send(sum.encode())

print("Connection closed from client")

#Close the connection with the client
conn.close()
```

Client

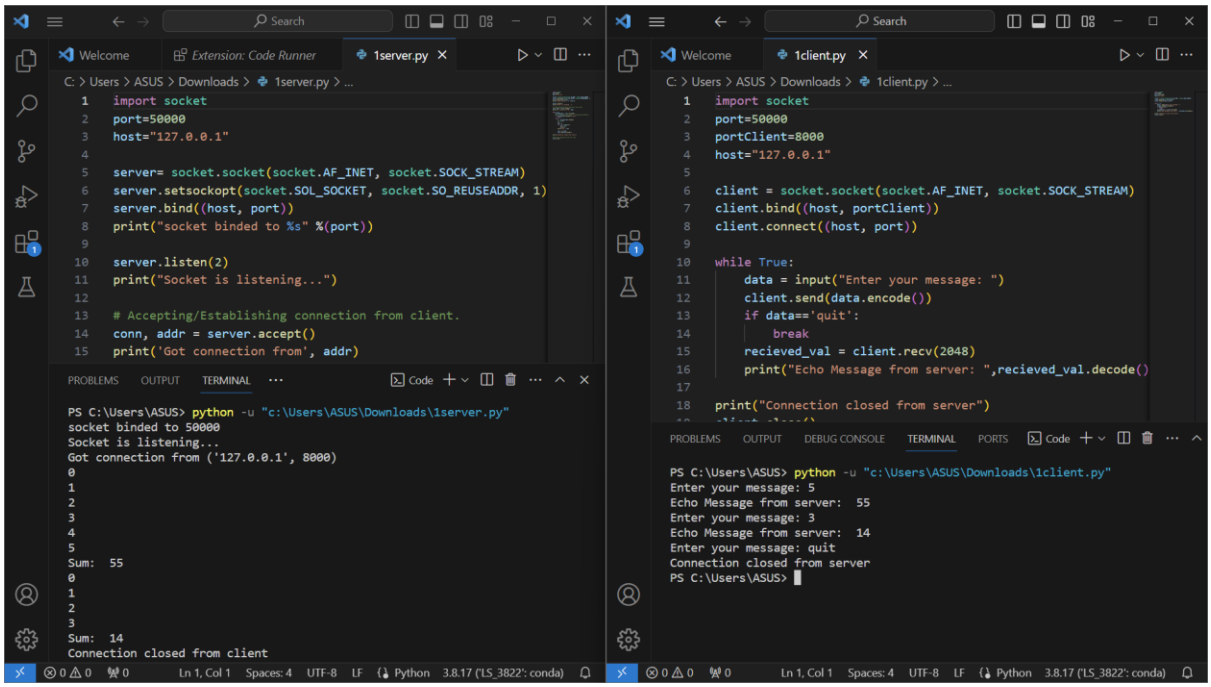
```
import socket
port=50000
portClient=8000
host="127.0.0.1"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.bind((host, portClient))
client.connect((host, port))

while True:
    data = input("Enter your message: ")
    client.send(data.encode())
    if data=='quit':
        break
    recieved_val = client.recv(2048)
    print("Echo Message from server: ",recieved_val.decode())

print("Connection closed from server")
client.close()
```

Results



Problem Statement 2

Write a TCP socket program (in C/C++/Java/Python) to establish connection between client and server. The client program will send a set of binary values to the server and the server program will return the number of 1s present in the data received. Client will display the value send by server. The communication between client and server will continue until client send ‘Quit’ message to the server.

Code

Server

```
import socket
port=50000
host="127.0.0.1"

server= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((host, port))
print("socket binded to %s" %(port))

server.listen(2)
print("Socket is listening...")

# Accepting/Establishing connection from client.
conn, addr = server.accept()
print('Got connection from', addr)

while True:
```

```
    recieved_data = conn.recv(2048)
    #print("Message from client: ",recieved_data.decode())
    if recieved_data.decode()=='quit':
        break
    else:
        n = recieved_data.decode()
        n = int(n)
        count = 0
        while (n):
            if n%2 == 1:
                count += 1
                n = n//10
            else:
                n = n//10
        print("No. of Ones: ",count)
        count = str(count)
        conn.send(count.encode())

print("Connection closed from client")

#Close the connection with the client
conn.close()
```

Client

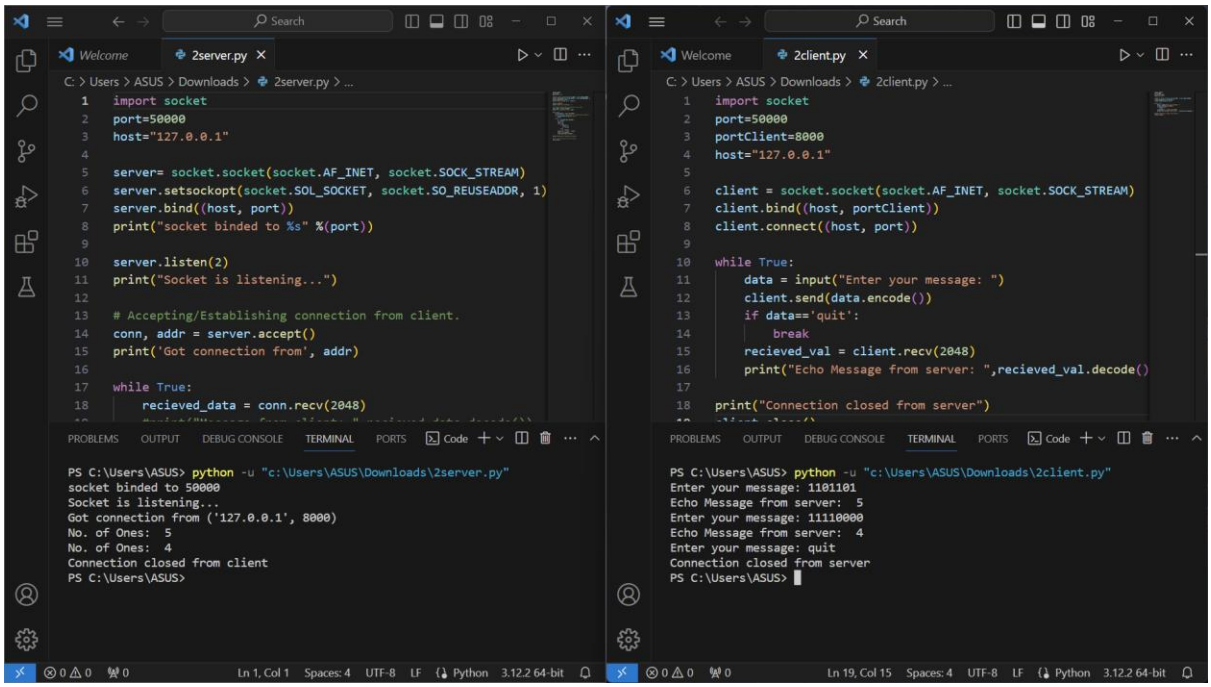
```
import socket
port=50000
portClient=8000
host="127.0.0.1"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.bind((host, portClient))
client.connect((host, port))

while True:
    data = input("Enter your message: ")
    client.send(data.encode())
    if data=='quit':
        break
    recieved_val = client.recv(2048)
    print("Echo Message from server: ",recieved_val.decode())

print("Connection closed from server")
client.close()
```

Results



Problem Statement 3

Write a TCP socket program (in C/C++/Java/Python) to establish connection between client and server. The client program will send a postfix expression to the server and the server program will return the result of the input expression. Server program will use a stack to evaluate the postfix expression. Client will display the value send by server. The communication between client and server will continue until client send ‘Quit’ message to the server.

Code

Server

```
import socket

def evaluate_postfix(expression):
    stack = []
    operators = set(['+', '-', '*', '/'])

    for char in expression:
        if char.isdigit():
            stack.append(int(char))
        elif char in operators:
            operand2 = stack.pop()
            operand1 = stack.pop()

            if char == '+':
                result = operand1 + operand2
```

```
            elif char == '-':
                result = operand1 - operand2
            elif char == '*':
                result = operand1 * operand2
            elif char == '/':
                result = operand1 / operand2

            stack.append(result)

    return stack[0]

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 12345))
    server_socket.listen(1)

    print("Server listening on port 12345...")

    while True:
        client_socket, addr = server_socket.accept()
        print(f"Connection established with {addr}")

        while True:
            data = client_socket.recv(1024).decode('utf-8')

            if data.lower() == 'quit':
                print("Connection closed by client.")
                client_socket.close()
                break

            result = evaluate_postfix(data)
            client_socket.send(str(result).encode('utf-8'))

if __name__ == "__main__":
    start_server()
```

Client

```
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 12345))

    while True:
        postfix_expression = input("Enter a postfix expression (or type 'Quit'
to exit): ")
```

```
        if postfix_expression.lower() == 'quit':
            client_socket.send('Quit'.encode('utf-8'))
            break

        client_socket.send(postfix_expression.encode('utf-8'))
        result = client_socket.recv(1024).decode('utf-8')
        print(f"Server response: Result of the expression: {result}")

    client_socket.close()

if __name__ == "__main__":
    start_client()
```

Results