Assignment 5

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 string to the server and the server program will check whether the string is a palindrome or not and send the response to the client accordingly. 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:

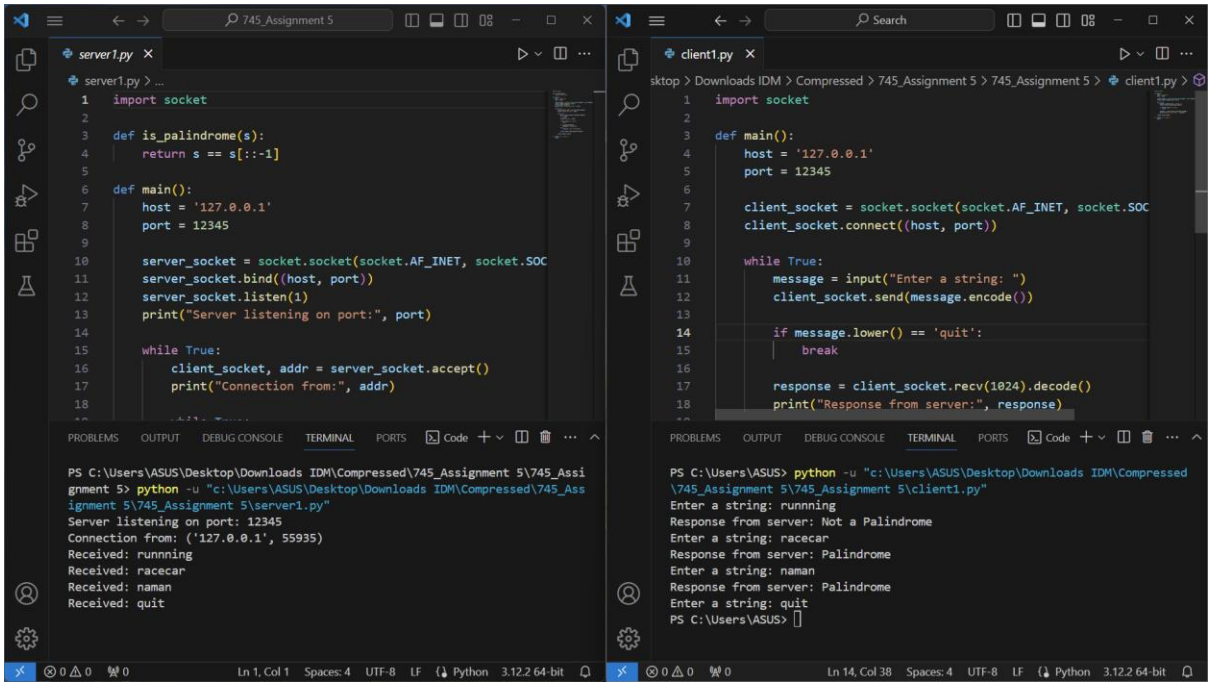
Client

```python
import socket

def main():
    host = '127.0.0.1'
    port = 12345

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    while True:
        message = input("Enter a string: ")
        client_socket.send(message.encode())

        if message.lower() == 'quit':
            break

        response = client_socket.recv(1024).decode()
        print("Response from server:", response)

    client_socket.close()

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

Server

```python
import socket

def is_palindrome(s):
    return s == s[::-1]

def main():
    host = '127.0.0.1'
    port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(1)
    print("Server listening on port:", port)

    while True:
        client_socket, addr = server_socket.accept()
        print("Connection from:", addr)

        while True:
            data = client_socket.recv(1024).decode()
            if not data:
                break
            print("Received:", data)

            if data.lower() == 'quit':
                break

            if is_palindrome(data):
                response = "Palindrome"
            else:
                response = "Not a Palindrome"

            client_socket.send(response.encode())

        client_socket.close()

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

Output



```
server1.py
import socket

def is_palindrome(s):
    return s == s[::-1]

def main():
    host = '127.0.0.1'
    port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOC
    server_socket.bind((host, port))
    server_socket.listen(1)
    print("Server listening on port:", port)

    while True:
        client_socket, addr = server_socket.accept()
        print("Connection from:", addr)
```

```
PS C:\Users\ASUS\Desktop\Downloads IDM\Compressed\745_Assignment 5\745_Assi
gnment 5> python -u "c:\Users\ASUS\Desktop\Downloads IDM\Compressed\745_Ass
ignment 5\745_Assignment 5\server1.py"
Server listening on port: 12345
Connection from: ('127.0.0.1', 55935)
Received: runnning
Received: racecar
Received: naman
Received: quit
```

```
client1.py
import socket

def main():
    host = '127.0.0.1'
    port = 12345

    client_socket = socket.socket(socket.AF_INET, socket.SOC
    client_socket.connect((host, port))

    while True:
        message = input("Enter a string: ")
        client_socket.send(message.encode())

        if message.lower() == 'quit':
            break

        response = client_socket.recv(1024).decode()
        print("Response from server:", response)
```

```
PS C:\Users\ASUS> python -u "c:\Users\ASUS\Desktop\Downloads IDM\Compressed
\745_Assignment 5\745_Assignment 5\client1.py"
Enter a string: runnning
Response from server: Not a Palindrome
Enter a string: racecar
Response from server: Palindrome
Enter a string: naman
Response from server: Palindrome
Enter a string: quit
PS C:\Users\ASUS>
```

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 string to the server and server program will generate the reverse of that string and send it back to the client. 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:

Client

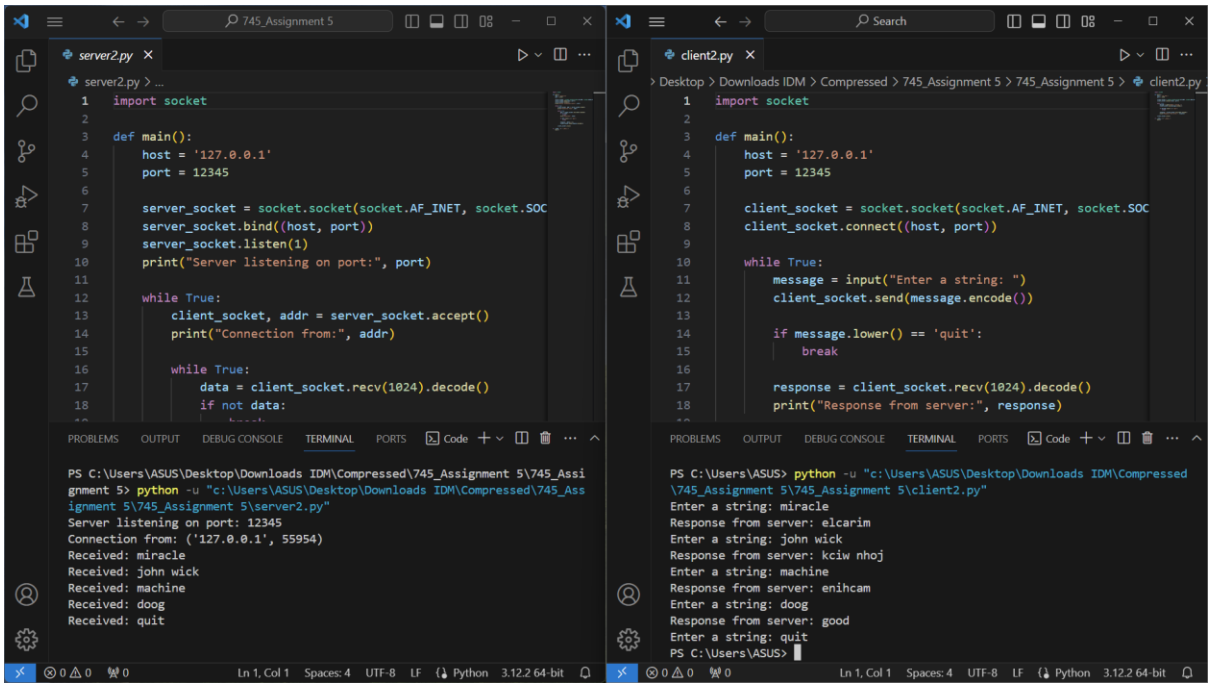
```
import socket

def main():
    host = '127.0.0.1'
    port = 12345

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    while True:
        message = input("Enter a string: ")
        client_socket.send(message.encode())

        if message.lower() == 'quit':
            break

        response = client_socket.recv(1024).decode()
        print("Response from server:", response)

    client_socket.close()

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

Server

```
import socket

def main():
    host = '127.0.0.1'
    port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(1)
    print("Server listening on port:", port)

    while True:
```

```
        client_socket, addr = server_socket.accept()
        print("Connection from:", addr)

        while True:
            data = client_socket.recv(1024).decode()
            if not data:
                break
            print("Received:", data)

            if data.lower() == 'quit':
                break

            response = data[::-1]
            client_socket.send(response.encode())

        client_socket.close()

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

Output



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 URL to the server and a depth up to which the web-crawler visits all the pages from the initial page. Server will use the URL, run a web crawler function up to the given depth to check all the URLs available through the input URL and send the list of those URLs to Client. Client will display the list send by server. The communication between client and server will continue until client send 'Quit' message to the server.

Code:

Client

```
import socket

def communicate_with_server():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 8888))

    while True:
        url = input("Enter a URL (or 'Quit' to exit): ")
        client_socket.send(url.encode('utf-8'))

        if url.lower() == 'quit':
            break

        depth = int(input("Enter the depth for web crawling: "))
        client_socket.send(str(depth).encode('utf-8'))

        response = client_socket.recv(4096).decode('utf-8')
        print(f"Server response:\n{response}")

    client_socket.close()

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

Server

```
import socket
from bs4 import BeautifulSoup
import requests

def web_crawler(url, current_depth, max_depth, visited_urls=set()):
    result_urls = []

    if current_depth > max_depth:
        return result_urls

    try:
```

```
        response = requests.get(url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')
            result_urls.append(url)

            for link in soup.find_all('a', href=True):
                next_url = link.get('href')
                if next_url.startswith(('http://', 'https://')):
                    if next_url not in visited_urls:
                        visited_urls.add(next_url)
                        result_urls.extend(web_crawler(next_url, current_depth
+ 1, max_depth, visited_urls))

    except requests.RequestException as e:
        print(f"Error while processing {url}: {e}")

    return result_urls

def handle_client(client_socket):
    while True:
        url = client_socket.recv(1024).decode('utf-8')
        if not url:
            break

        if url.lower() == 'quit':
            break

        depth = int(client_socket.recv(1024).decode('utf-8'))
        result_urls = web_crawler(url, 1, depth)

        # Send the list of URLs to the client
        response = '\n'.join(result_urls)
        client_socket.send(response.encode('utf-8'))

    client_socket.close()

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 8888))
    server_socket.listen(5)
    print("Server listening on port 8888...")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Accepted connection from {client_address}")
        handle_client(client_socket)

if __name__ == "__main__":
```

```
start_server()
```

Output: