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**

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 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"

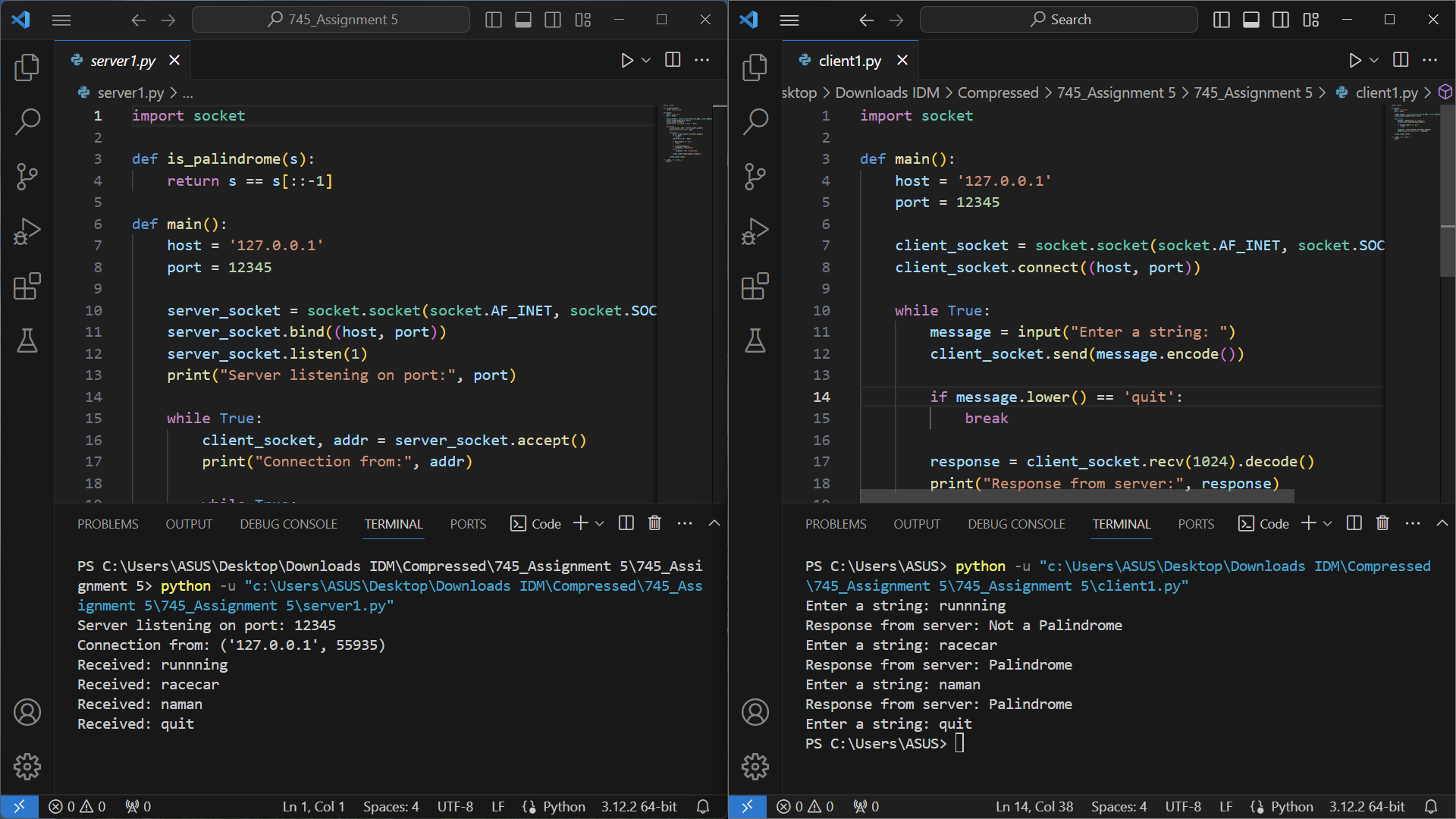
            client\_socket.send(response.encode())

        client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output



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]

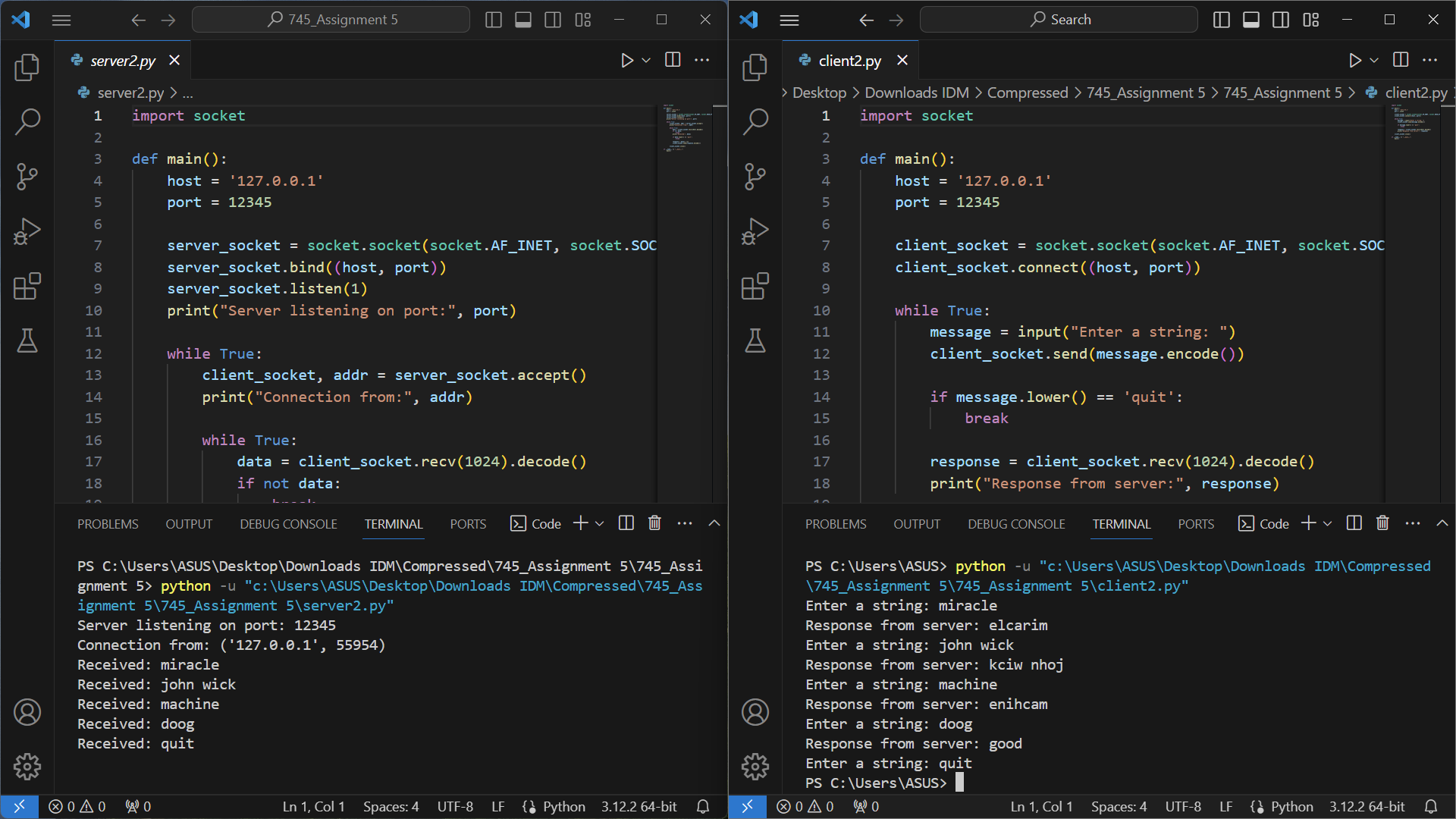
            client\_socket.send(response.encode())

        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:

