Computer Networks

Assignment – 8

Name: Anik Barury
Roll: CSE22017
Reg: 871

Q.To implement a client-server program using TCP/UDP sockets in Python for handling multiple clients on the server with multithreading. The program should allow multiple clients to connect to the server simultaneously and exchange messages. Each client connection should be handled in a separate thread.

Server.py

```
import socket
import threading

clients = []
def handle_client(client_socket):
    clients.append(client_socket)
    try:
        while True:
            message = client_socket.recv(1024).decode('utf-8')

            if not message or message.lower() == 'exit':
                print("Client disconnected")
                break

            print(f"Received from client: {message}")
            for client in clients:
                try:
                    client.send(f"Message from server: {message}".encode('utf-8'))
                except:
                    clients.remove(client)
    except Exception as e:
        print(f"Error: {e}")
    finally:
        clients.remove(client_socket)
        client_socket.close()

def start_server(host, port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"Server listening on {host}:{port}...")
```

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

        client_thread = threading.Thread(target=handle_client, args=(client_socket,))
        client_thread.start()

if __name__ == "__main__":
    host = '127.0.0.1'
    port = 12345

    start_server(host, port)
```

Client.py: (1+2)

```
import socket
import threading

def listen_for_messages(client_socket):
    try:
        while True:
            message = client_socket.recv(1024).decode('utf-8')
            if message:
                print(f"Received from server: {message}")
            else:
                break
    except Exception as e:
        print(f"Error: {e}")
    finally:
        client_socket.close()

def send_message_to_server(client_socket):
    try:
        while True:
            message = input("Enter message to send to the server (or 'exit' to disconnect): ")

            if message.lower() == 'exit':
                print("Disconnecting from server...")
                client_socket.send(message.encode('utf-8'))
                break
            client_socket.send(message.encode('utf-8'))
    except Exception as e:
        print(f"Error: {e}")
    finally:
        client_socket.close()

def start_client(host, port):
```

```
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    client_socket.connect((host, port))

    listen_thread = threading.Thread(target=listen_for_messages, args=(client_socket,))
    listen_thread.daemon = True
    listen_thread.start()

    send_message_to_server(client_socket)

if __name__ == "__main__":
    host = '127.0.0.1'
    port = 12345

    start_client(host, port)
```

OUTPUT: