END-END COMMUNICATION AT TRANSPORT LAYER

AIM :(a)
Implement echo client server using TCP/UDP sockets .

CODE :

```
import socket
def start_tcp_echo_server():
        serversocket = socket. socket (socket.AF_INET,
                              socket. sock_STREAM
        server socket = bind ('local host', 12345)
        serversocket. listen(1)
       print ("server listening on port 12345...")
     while true:
         client socket , client address = serversocket. accept ()
         print(f"Connection established with {client_address}")

         try:
           data = client-socket . recv (1024)
           if data:
               print(f"Received : {data. decode()}")
               clientsocket. sendall (data)

         finally:
              clientsocket. close ()
  if_name__ == "main" :
          start-tcp-echo server ()

  def tcp-echo -client ():
        client_socket = socket. socket (socket .AF_INET,
                              socket . SOCK_STREAM)
        client_socket. connect(('localhost', 12345))
        message = "Hello, server"
```

```
client_socket.sendall(message.encode())
response = client_socket.recv(1024)
print(f"Received from server: {response.decode()}")
client_socket.close()
```

INPUT:

Input to server : "Hello, Server"

OUTPUT:

```
Server listening on port 12345...
Connection established with ('127.0.0.1', 54321)
Output from server: "Hello, Server"
```

RESULT:

Echo client server using TCP/UDP implemented successfully

[illegible]

AIM : (b)

Implement chat client server using TCP/UDP Sockets.

CODE :

```
import socket
import threading
clients = []
def broadcast (message, client_socket):
    for client in clients :
        if client != client_socket
        try:
          client.send (message)
        except:
          clients. remove (client)
def handle_client (client_socket):
   while true :
      try:
        message = clientsocket. recv(1024)
        if message :
            broadcast (message , client_socket)
        else :
            break
      except:
            break
def start_chat_server():
    serversocket = socket. socket (Socket . AF_INET , socket.
                          SOCK_STREAM)
    server_socket. bind ('localhost', 12346)
    server_socket . listen (5)
   while True :
     client_socket, client_address = server_socket .accept()
     clients. append (client_socket)
```

```
def receive_message (client_socket):
    while True:
        try:
            message = client_socket. recv (1024)
            if message:
                print(f"\n{message. decode()}")
            else:
                break
        except:
            break

def start_chat_client ():
    clientsocket = socket.socket (socket. AF_INET,
                          socket. SOCK_STREAM)
    threading. Thread (target = receive_messages,
    args = (client_socket,)). start()
    while True:
        message = input()
        client_socket. send (message. encode())
```

INPUT:

```
Enter message: Hello
```

OUTPUT:

```
Received from other clients: Hello
```

RESULT:

Chat client server using TCP/UDP implemented successfully

14/X 9/10