Aim: To implement a basic **TCP Client–Server communication** program in C where the server listens on a port, accepts a connection, receives a message from the client, and sends a reply back.

To understand network programming concepts such as **socket creation, binding, listening, accepting and connecting**.

THEORY

1. Socket Programming
A **socket** is an endpoint for communication. TCP sockets provide **reliable, connection-oriented** byte streams.

2. Important System Calls
- **socket()** → creates a socket (TCP/UDP).
- **bind()** → assigns an IP address + port number to the socket (server).
- **listen()** → puts the socket into passive mode, waiting for clients.
- **accept()** → accepts a client connection and creates a new socket for communication.
- **connect()** → client connects to the server.
- **send() / write()** → sends data.
- **recv() / read()** → receives data.
- **close()** → closes the socket.

3. Client–Server Model
- The **server** runs first, binds to a specific port, and waits.
- The **client** connects to the server using the server IP + port.
- Messages are exchanged using TCP.
- Both sides close the connection after communication.

4. LAN vs Same Device Communication
- If client/server are on **same device**, use IP **127.0.0.1 (loopback)**.
- If on **different devices**, use the **actual LAN IP of the server** (e.g., 192.168.x.x).

PROCEDURE

A. Server Side
1. Create a socket using socket(AF_INET, SOCK_STREAM, 0).
2. Set socket options using setsockopt() to allow reuse of the port.
3. Fill the server address structure with:
   o Address family → AF_INET
   o Port → 8080
   o IP → INADDR_ANY
4. Bind the socket to the IP/port using bind().
5. Call listen() to wait for incoming clients.
6. Accept an incoming connection using accept().
7. Read the message sent by client.
8. Print the message.

9. Send a reply back: "Hello from server".
10. Close the connection socket.

B. Client Side
1. Create a socket using socket().
2. Fill server IP and port in the structure.
   - For local → 127.0.0.1
   - For remote → Replace with server's LAN IP.
3. Call connect() to establish connection with server.
4. Send "Hello from client".
5. Read server's reply and print it.
6. Close the socket.

C. Compilation (Linux GCC)
gcc server.c -o server
gcc client.c -o client

D. Execution
1. Run server first:
2. ./server
3. Run client:
4. ./client

RESULT
1. The client successfully established a TCP connection with the server.
2. The client sent the message **"Hello from client"** to the server.
3. The server received and displayed the message.
4. The server sent back **"Hello from server"**, which the client received and printed.
5. The program demonstrated correct functioning of **TCP socket programming** and **two-way communication**.

CONCLUSION
The experiment was successfully performed.
We learned how TCP client–server communication works using socket programming in C.
We implemented:
- socket creation
- binding
- listening
- accepting
- connecting
- sending and receiving messages

The practical demonstrated how reliable two-way communication occurs over a network using the TCP protocol.