//Djikstra

```c
#include <stdio.h>
#define INFINITY 9999
#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start);

void Dijkstra(int Graph[MAX][MAX], int n, int start)
{
    int cost[MAX][MAX], distance[MAX], pred[MAX];
    int visited[MAX], count, mindistance, nextnode, i, j;

    // Creating cost matrix
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (Graph[i][j] == 0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = Graph[i][j];

    for (i = 0; i < n; i++)
    {
        distance[i] = cost[start][i];
        pred[i] = start;
        visited[i] = 0;
    }

    distance[start] = 0;
    visited[start] = 1;
    count = 1;

    while (count < n - 1)
    {
        mindistance = INFINITY;

        for (i = 0; i < n; i++)
            if (distance[i] < mindistance && !visited[i])
            {
                mindistance = distance[i];
                nextnode = i;
            }

        visited[nextnode] = 1;
        for (i = 0; i < n; i++)
            if (!visited[i])
                if (mindistance + cost[nextnode][i] < distance[i])
                {
```

```
                    distance[i] = mindistance + cost[nextnode][i];
                    pred[i] = nextnode;
                }
        count++;
    }

    // Printing the distance
    for (i = 0; i < n; i++)
        if (i != start)
        {
            printf("\nDistance from source to %d: %d", i, distance[i]);
        }
}
int main()
{
    int graph[MAX][MAX], i, j, n, u;
    scanf("%d",&n);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &graph[i][j]);
        }
    }
    u = 0;
    Dijkstra(graph, n, u);

    return 0;
}
```

//UDP client

```
import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;


public class udpBaseClient_2

{

        public static void main(String args[]) throws IOException

        {
```

```
        Scanner sc = new Scanner(System.in);

        // Step 1:Create the socket object for
        // carrying the data.
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        // loop while user not enters "bye"
        while (true)
        {
            String inp = sc.nextLine();

            // convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2 : Create the datagramPacket for sending
            // the data.
            DatagramPacket DpSend =
                new DatagramPacket(buf, buf.length, ip, 1234);

            // Step 3 : invoke the send call to actually send
            // the data.
            ds.send(DpSend);

            // break the loop if user enters "bye"
            if (inp.equals("bye"))
                break;
        }
}
```

```
}

//UDP server
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

public class udpBaseServer_2
{
	public static void main(String[] args) throws IOException
	{
		// Step 1 : Create a socket to listen at port 1234
		DatagramSocket ds = new DatagramSocket(1234);
		byte[] receive = new byte[65535];

		DatagramPacket DpReceive = null;
		while (true)
		{

			// Step 2 : create a DatgramPacket to receive the data.
			DpReceive = new DatagramPacket(receive, receive.length);

			// Step 3 : revieve the data in byte buffer.
			ds.receive(DpReceive);

			System.out.println("Client:-" + data(receive));

			// Exit the server if the client sends "bye"
			if (data(receive).toString().equals("bye"))
```

```
                        {
                                System.out.println("Client sent bye.....EXITING");
                                break;
                        }

                        // Clear the buffer after every message.
                        receive = new byte[65535];
                }
        }

        // A utility method to convert the byte array
        // data into a string representation.
        public static StringBuilder data(byte[] a)
        {
                if (a == null)
                        return null;
                StringBuilder ret = new StringBuilder();
                int i = 0;
                while (a[i] != 0)
                {
                        ret.append((char) a[i]);
                        i++;
                }
                return ret;
        }
}
```

// TCP Client

```
import java.io.*;
import java.net.*;
```

```
public class Client {
	// initialize socket and input output streams
	private Socket socket = null;
	private DataInputStream input = null;
	private DataOutputStream out = null;

	// constructor to put ip address and port
	public Client(String address, int port)
	{
		// establish a connection
		try {
			socket = new Socket(address, port);
			System.out.println("Connected");

			// takes input from terminal
			input = new DataInputStream(System.in);

			// sends output to the socket
			out = new DataOutputStream(
				socket.getOutputStream());
		}
		catch (UnknownHostException u) {
			System.out.println(u);
			return;
		}
		catch (IOException i) {
			System.out.println(i);
			return;
		}
```

```
            // string to read message from input
            String line = "";

            // keep reading until "Over" is input
            while (!line.equals("Over")) {
                try {
                    line = input.readLine();
                    out.writeUTF(line);
                }
                catch (IOException i) {
                    System.out.println(i);
                }
            }

            // close the connection
            try {
                input.close();
                out.close();
                socket.close();
            }
            catch (IOException i) {
                System.out.println(i);
            }
        }

        public static void main(String args[])
        {
            Client client = new Client("127.0.0.1", 5000);
        }
}
```

```
// TCP Server
import java.net.*;
import java.io.*;

public class Server
{
	//initialize socket and input stream
	private Socket		socket = null;
	private ServerSocket server = null;
	private DataInputStream in	= null;

	// constructor with port
	public Server(int port)
	{
		// starts server and waits for a connection
		try
		{
			server = new ServerSocket(port);
			System.out.println("Server started");

			System.out.println("Waiting for a client ...");

			socket = server.accept();
			System.out.println("Client accepted");

			// takes input from the client socket
			in = new DataInputStream(
				new BufferedInputStream(socket.getInputStream()));

			String line = "";
```

```
                    // reads message from client until "Over" is sent
                    while (!line.equals("Over"))
                    {
                        try
                        {
                            line = in.readUTF();
                            System.out.println(line);

                        }
                        catch(IOException i)
                        {
                            System.out.println(i);
                        }
                    }
                    System.out.println("Closing connection");

                    // close connection
                    socket.close();
                    in.close();
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
        }

        public static void main(String args[])
        {
            Server server = new Server(5000);
        }
}
```

```
// Server2 class that
// receives data and sends data

import java.io.*;
import java.net.*;

class Server2 {

	public static void main(String args[])
		throws Exception
	{

		// Create server Socket
		ServerSocket ss = new ServerSocket(888);

		// connect it to client socket
		Socket s = ss.accept();
		System.out.println("Connection established");

		// to send data to the client
		PrintStream ps
			= new PrintStream(s.getOutputStream());

		// to read data coming from the client
		BufferedReader br
			= new BufferedReader(
				new InputStreamReader(
					s.getInputStream()));

		// to read data from the keyboard
		BufferedReader kb
```

```
            = new BufferedReader(
                        new InputStreamReader(System.in));

// server executes continuously
while (true) {

            String str, str1;

            // repeat as long as the client
            // does not send a null string

            // read from client
            while ((str = br.readLine()) != null) {
                        System.out.println(str);
                        str1 = kb.readLine();

                        // send to client
                        ps.println(str1);
            }

            // close connection
            ps.close();
            br.close();
            kb.close();
            ss.close();
            s.close();

            // terminate application
            System.exit(0);

} // end of while
```

```
        }
}
```

// Client2 class that

// sends data and receives also

```
import java.io.*;
import java.net.*;

class Client2 {

        public static void main(String args[])
                throws Exception
        {

                // Create client socket
                Socket s = new Socket("localhost", 888);

                // to send data to the server
                DataOutputStream dos
                        = new DataOutputStream(
                                s.getOutputStream());

                // to read data coming from the server
                BufferedReader br
                        = new BufferedReader(
                                new InputStreamReader(
                                        s.getInputStream()));

                // to read data from the keyboard
```

```
                BufferedReader kb
                        = new BufferedReader(
                                new InputStreamReader(System.in));
                String str, str1;

                // repeat as long as exit
                // is not typed at client
                while (!(str = kb.readLine()).equals("exit")) {

                        // send to the server
                        dos.writeBytes(str + "\n");

                        // receive from the server
                        str1 = br.readLine();

                        System.out.println(str1);
                }

                // close connection.
                dos.close();
                br.close();
                kb.close();
                s.close();
        }
}
```