1.

#include<stdio.h>

#include<string.h>

#define N strlen(gen\_poly)

char data[28];

char check\_value[28];

char gen\_poly[10];

int data\_length,i,j;

void XOR(){

for(j = 1;j < N; j++)

check\_value[j] = (( check\_value[j] == gen\_poly[j])?'0':'1');

}

void receiver(){

printf("Enter the received data: ");

scanf("%s", data);

printf("\n-----------------------------\n");

printf("Data received: %s", data);

crc();

for(i=0;(i<N-1) && (check\_value[i]!='1');i++);

if(i<N-1)

printf("\nError detected\n\n");

else

printf("\nNo error detected\n\n");

}

void crc(){

for(i=0;i<N;i++)

check\_value[i]=data[i];

do{

if(check\_value[0]=='1')

XOR();

for(j=0;j<N-1;j++)

check\_value[j]=check\_value[j+1];

check\_value[j]=data[i++];

}while(i<=data\_length+N-1);

}

int main()

{

printf("\nEnter data to be transmitted: ");

scanf("%s",data);

printf("\n Enter the Generating polynomial: ");

scanf("%s",gen\_poly);

data\_length=strlen(data);

for(i=data\_length;i<data\_length+N-1;i++)

data[i]='0';

printf("\n----------------------------------------");

printf("\n Data padded with n-1 zeros : %s",data);

printf("\n----------------------------------------");

crc();

printf("\nCRC or Check value is : %s",check\_value);

for(i=data\_length;i<data\_length+N-1;i++)

data[i]=check\_value[i-data\_length];

printf("\n----------------------------------------");

printf("\n Final data to be sent : %s",data);

printf("\n----------------------------------------\n");

receiver();

return 0;

}

2.

#include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()

{

int G[MAX][MAX],i,j,n,u;

printf("Enter no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&G[i][j]);

printf("\nEnter the starting node:");

scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

for(i=0;i<n;i++)

for(j=0;j<n;j++)

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

for(i=0;i<n;i++)

{

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=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++;

}

for(i=0;i<n;i++)

if(i!=startnode)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

j=pred[j];

printf("<-%d",j);

}while(j!=startnode);

}

}

3.

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int costmat[20][20];

int nodes,i,j,k,count=0;

printf("\nEnter the number of nodes : ");

scanf("%d",&nodes);

printf("\nEnter the cost matrix :\n");

for(i=0;i<nodes;i++)

{

for(j=0;j<nodes;j++)

{

scanf("%d",&costmat[i][j]);

costmat[i][i]=0;

rt[i].dist[j]=costmat[i][j];

rt[i].from[j]=j;

}

}

do

{

count=0;

for(i=0;i<nodes;i++)

for(j=0;j<nodes;j++)

for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];

rt[i].from[j]=k;

count++;

}

}while(count!=0);

for(i=0;i<nodes;i++)

{

printf("\n\n For router %d\n",i+1);

for(j=0;j<nodes;j++)

{

printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);

}

}

return 0;

}

4.

#include <stdio.h>

int main() {

int no\_of\_queries, storage, output\_pkt\_size;

int input\_pkt\_size, bucket\_size, size\_left;

printf("Enter the total number of times the bucket's content is checked: ");

scanf("%d", &no\_of\_queries);

printf("Enter the total number of packets that can be accommodated in the bucket: ");

scanf("%d", &bucket\_size);

printf("Enter the number of packets that enter the bucket at a time: ");

scanf("%d", &input\_pkt\_size);

printf("Enter the number of packets that exit the bucket at a time: ");

scanf("%d", &output\_pkt\_size);

storage = 0;

for (int i = 0; i < no\_of\_queries; i++) {

size\_left = bucket\_size - storage;

if (input\_pkt\_size <= size\_left) {

storage += input\_pkt\_size;

} else {

printf("Packet loss = %d\n", input\_pkt\_size);

}

printf("Buffer size = %d out of bucket size = %d\n", storage, bucket\_size);

storage -= output\_pkt\_size;

}

return 0;

}

5.

#include <stdio.h>

int main() {

int token\_bucket\_capacity, token\_rate, packet\_size, num\_packets;

printf("Enter token bucket capacity: ");

scanf("%d", &token\_bucket\_capacity);

printf("Enter token rate (tokens per second): ");

scanf("%d", &token\_rate);

printf("Enter packet size: ");

scanf("%d", &packet\_size);

printf("Enter the number of packets to send: ");

scanf("%d", &num\_packets);

int tokens = 0;

for (int i = 0; i < num\_packets; i++) {

tokens += token\_rate;

if (tokens >= packet\_size) {

printf("Sending packet %d (Size: %d bytes)\n", i + 1, packet\_size);

tokens -= packet\_size;

} else {

printf("Dropping packet %d (Size: %d bytes)\n", i + 1, packet\_size);

}

}

return 0;

}

6.IMPLEMENTATION OF TCP CLIENT AND SERVER PROGRAM

// A Java program for a Client

import java.net.\*;

import java.io.\*;

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);

}

catch(IOException i)

{

System.out.println(i);

}

// 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);

}

}

SERVER PROGRAM

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);

}

}

7.IMPLEMENTATION OF UDP CLIENT AND SERVER

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;

}

}