1. Character Stuffing

1. **Aim** : Write a program for Character Stuffing

Procedure:

1. Start
2. Append DLE STX at the beginning of the string
3. Check the data if character is present; if character DLE is present in the string (example DOODLE) insert another DLE in the string (ex: DOODLEDLE)
4. Transmit DLE ETXat the end of the string
5. Display the string
6. Stop

Program

```
import java.util.*;
class Char
{
   public static void main(String r[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter number of characters: ");
      int n=sc.nextInt();
      String in[]=new String[n];
      for(int i=0;i<n;i++)
      {
         in[i]=sc.next();
      }
      for(int i= 0;i<n;i++)
      {
         if(in[i].equals("dle"))
         {
            in[i]="dle dle";
         }
      }
      System.out.println("Transmitted message is: ");
      System.out.print(" dle stx ");
      for(int i=0;i<n;i++)
      {
         System.out.print(in[i]+" ");
      }
      System.out.println(" dle etx ");
   }
}
```

Output

Sample output:

```
Enter number of characters:
5
a b dle dle d e
Transmitted message is:
dle stxa b dle dle dle dle d dle etx
```

2. **SELECTIVE REPERT PROTOCOL SERVER**

2. **Aim**: Write a Program for Selective Repeat Protocol

Procedure:

1. Start.
2. Establish connection (recommended UDP)
3. Accept the window size from the client(should be <=40)
4. Accept the packets from the network layer.
5. Calculate the total frames/windows required.
6. Send the details to the client(totalpackets,totalframes.)
7. Initialise the transmit buffer.
8. Built the frame/window depending on the windowsize.
9. Transmit the frame.
10. Wait for the acknowledgement frame.
11. Check for the acknowledgement of each packet and repeat the process for the packet for which the negative acknowledgement isreceived. Else continue as usual.
12. Increment the frame count and repeat steps 7 to 12 until all packets are transmitted.
13. Close the connection.
14. Stop.

Program:

```
import java.io.*;
import java.net.*;
import java.util.LinkedList;
public class SelRserver {
   public static void main(String args[]){
      try {
         //Create Socket at Server Side
         ServerSocket s=new ServerSocket(95);
         Socket con =s.accept();
         System.out.println("Server Selective Repeat Started...");
```

```
        BufferedReader fromclient = new BufferedReader(new
InputStreamReader(con.getInputStream()));
        DataOutputStream toclient= new DataOutputStream(con.getOutputStream());
        OutputStream output = con.getOutputStream();

        LinkedList list = new LinkedList();

        String f = fromclient.readLine();

        int nf=Integer.parseInt(f);
        System.out.println("Number of packets = "+nf);

        int lost_ack=0,rndnum=1,lost=0;
        String rndm=fromclient.readLine();
        rndnum= Integer.parseInt(rndm);

        String flow = fromclient.readLine();


        if(flow.contains("n") || flow.contains("N")){
           for(int i=0;i
              if(i==rndnum) {
                 System.out.println("lost packet is"+i);
                 list.add(i,"lost");
                 lost=i;
                 continue;
              }

              String n=fromclient.readLine();

              int num=Integer.parseInt(n);
              String packet=fromclient.readLine();
              list.add(num,packet);

              if(list.contains("lost")){

                 lost_ack= list.indexOf("lost");
                 toclient.writeBytes(lost_ack+"\n");
                 num=lost_ack;


                 if(i>rndnum){
                    System.out.println("sending ack"+i);
                    toclient.flush();
                    toclient.writeBytes(i+"\n");
```

```
                toclient.flush();
            }else
            {
                System.out.println("sending ack"+num);
                toclient.flush();
                toclient.writeBytes(num+"\n");
                toclient.flush();
            }
        }else
        {
            System.out.println("sending ack"+num);
            toclient.flush();
            toclient.writeBytes(num+"\n");
            toclient.flush();
        }
    }
    System.out.println(list);
    String n=fromclient.readLine();
    int num=Integer.parseInt(n);
    String packet=fromclient.readLine();
    System.out.println("received packet"+num+" = "+packet);
    list.set(lost,packet);
    System.out.println(list);
    toclient.flush();
    int ack=num;

    //Send the acknowledgment for lost packet,once it is received from client again.
    if(list.contains("lost")) {
        lost_ack= list.indexOf("lost");
        System.out.println("Sending lost Ack"+lost_ack);
        toclient.writeBytes(lost_ack+"\n");
    } else {
        System.out.println("Sending Ack"+ack);
        toclient.writeBytes(ack+"\n");
    }
}else{
    for(int i=0;i                String n=fromclient.readLine();
        int num=Integer.parseInt(n);
        String packet=fromclient.readLine();
        list.add(num,packet);
        System.out.println("received packet"+num+" = "+packet);
        System.out.println("sending ack"+i);
        toclient.flush();
        toclient.writeBytes(i+"\n");
        toclient.flush();
```

```
            }
            System.out.println("received the following packets"+list);
         }
         //Close connections with client
         fromclient.close();
         output.close();
         con.close();
         s.close();
      }catch (Exception e) {
         System.out.println("\n\n ERROR"+e);
      }
   }
}
```

/****** **Selective Repeat Client** *******/

```
import java.io.*;
import java.net.*;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.LinkedList;
import java.util.*;

public class SelRClient {

@SuppressWarnings("static-access")
   public static void main(String args[]) {

   long start = System.currentTimeMillis();
   try
   {
      //Create Socket at client side
      Socket SRClient =new Socket("localhost",95);

      //Receive the input from console
      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
      DataOutputStream toSrServer= new DataOutputStream(SRClient.getOutputStream());
      BufferedReader fromSrServer = new BufferedReader(new
InputStreamReader(SRClient.getInputStream()));

      //Create linked list to display and store the packets
      LinkedList list = new LinkedList();
```

```
System.out.println("We maintained window size as 4,so for better performance evaluation kindly enter atleast 15 packets in input \n");

Random rndNumbers = new Random();
int rndNumber = 0;

System.out.println("Enter number of packets to be transmitted");

String n=in.readLine();
int nf=Integer.parseInt(n);
toSrServer.writeBytes(nf+"\n");
int temp=nf-1;

//Generate a random number so that particular packet is lost in demo
rndNumber = rndNumbers.nextInt(temp);

for(int i=0;i           System.out.println("Enter packet"+i+"to be transmitted");
   String packet=in.readLine();
   list.add(packet);
}
System.out.println("\nYou have entered following data "+list+"\n");

toSrServer.writeBytes(rndNumber+"\n");

System.out.println("Please enter Y for  successful sending or N for packet loss");
String flow = in.readLine();

toSrServer.writeBytes(flow+"\n");

 int w=4,wind=0,p=0,q=0;

//While sending packet ignore the random number generated packet.
 if(flow.contains("n") || flow.contains("N")){
    for(int i=0;i               p=i;
       wind = windowSize(p,q);

      if((i==rndNumber)){

         System.out.println("Packet lost is"+i);
         q=w;
         wind = windowSize(p,q);
         wind++;
         q=wind;
         continue;
```

```
        } else {
            String str=list.get(i);
            toSrServer.writeBytes(i+"\n");
            toSrServer.writeBytes(str+"\n");
        }

        wind++;
        q=wind;

        String ak = fromSrServer.readLine();
        int ack = Integer.parseInt(ak);
        ack=i;
        System.out.println("Ack received for "+ack);
    }

    //Re-send the lost packet
    System.out.println("Resending packet"+rndNumber);
    String str=list.get(rndNumber);

    toSrServer.writeBytes(rndNumber+"\n");
    toSrServer.writeBytes(str+"\n");
    toSrServer.flush();

//Timer set for expected acknowledgment
    Thread.currentThread().sleep(2000);

    String as = fromSrServer.readLine();

    System.out.println("Recieved ack no"+rndNumber);
    int aa = Integer.parseInt(as);
    if (aa != (list.size() -1))
        System.out.println("Ack received for "+rndNumber);
 }else
 {
     for(int i=0;i                p=i;
         wind = windowSize(p,q);
        String str=list.get(i);
         toSrServer.writeBytes(i+"\n");
         toSrServer.writeBytes(str+"\n");
         wind++;
        q=wind;
         String ak = fromSrServer.readLine();
        int ack = Integer.parseInt(ak);
        ack=i;
        System.out.println("Ack received for "+ack);
```

```
            }
         }
        //Close socket and server connections
        fromSrServer.close();
        toSrServer.close();
        SRClient.close();
    }catch (Exception e){
        System.out.println("\n\n ERROR"+e);
    }

    //Calculate performance
    long end = System.currentTimeMillis();
    NumberFormat formatter = new DecimalFormat("#0.00000");
    System.out.println("Execution time is " + formatter.format((end - start) / 1000d) + "
seconds");

    }
    public static int windowSize(int p,int wind) throws InterruptedException
    {
        int i=p,w=i+4,windw=wind;
        LinkedList < Integer > list1 = new LinkedList < Integer > ();
        if(i==0 || windw==4){
            System.out.println("\nBased on the window size, packets to be sent ");
            for(int q=i;q               windw=0;
                list1.add(q);

            }
            System.out.println(list1);
            if(windw==0){
                Thread.currentThread();
                Thread.sleep(2000);
            }
        }
        return windw;
    }
}
```

3.Bit Stuffing

3. **Aim**: Write a Program for Bit Stuffing

Procedure:

1.Start
2. Initialize the array for transmitted stream with the special bit pattern
0111 1110 which indicates the beginning of the frame.
3. Get the bit stream to be transmitted in to the array.
4. Check for five consecutive ones and if they occur, stuff a bit 0

5. Display the data transmitted as it appears on the data line after appending 0111 1110 at the end
6. For de−stuffing, copy the transmitted data to another array after detecting the stuffed bits
7. Display the received bit stream
8. Stop

Program:

```
import java.util.*;
public class SEFBS
        {
    public static void main(String[] args)
                    {
        System.out.print("Enter the Binary message: ");
        Scanner sn=new Scanner(System.in);
        String data = sn.nextLine();
        String res = new String();
        String out=new String();
        int counter = 0;
        for(int i=0;i<data.length();i++)
         {

           if (data.charAt(i)!='1' && data.charAt(i)!='0')
               {
                  System.out.println("Enter only Binary values!!!");
                  return;
               }
           if(data.charAt(i) == '1')
               {
                  counter++;
                  res = res + data.charAt(i);
               }
           else
               {
                  res = res + data.charAt(i);
                  counter = 0;
               }
           if(counter == 5)
               {
                  res = res + '0';
                  counter = 0;
               }
         }
       String inc="01111110"+res+"01111110";
       System.out.println("The Message to be transfered: " +inc);
       System.out.println("Seding Message....");
       counter=0;
```

```
        for(int i=0;i<res.length();i++)
           {

             if(res.charAt(i) == '1')
                {

                   counter++;
                   out = out + res.charAt(i);

                }
             else
                {
                    out = out + res.charAt(i);
                    counter = 0;
                }
            if(counter == 5)
                {
                    if((i+2)!=res.length())
                    out = out + res.charAt(i+2);
                    else
                    out=out + '1';
                    i=i+2;
                    counter = 1;
                }
           }

        System.out.println("Message Recevied...Successfully!!!");
        System.out.println("The Destuffed Message is: "+out);
    }
}
```

Output:

```
Case 1:
Enter the Binary message: 0111111110
The Message to be transferred: 01111110011111101110011111110
Sending Message....
Message Received...Successfully!!!
The Destuffed Message is: 0111111110

Case 2:
Enter the Binary message: 01111110
The Message to be transferred: 0111111001111101001111110
Sending Message....
Message Received...Successfully!!!
The Destuffed Message is: 01111110
```

4.Go Back N Protocol

4. **Aim**: Write a Program for Go Back N protocol
   Procedure:

N = window size

Sn = sequence number

Sb = sequence base

Sm = sequence max

ack = ack number

nack = first non-acknowledged

Receiver:

Do the following forever:

Randomly accept or reject packet

If the packet received and the packet is error free

Accept packet

Send a positive ack for packet

Else

Refuse packet

Send a negative ack for packet

Sender:

Sb = 0

Sm = N – 1

ack = 0

Repeat the following steps forever:

Send packet with ack

If positively ack is recieved:

ack++

Transmit a packet where Sb <= ack <= Sm.

packets are transmitted in order

Else

Enqueue the nack into the queue

//check if last packet in the window is sent

if(ack==Sm)

if(queue is not empty)

// start from the first nack packet

nack = queue.front();

empty the queue

ack = nack

Sm = Sm + (ack − Sb)

Sb = ack

Programme:

```
/*Server Program*/

import java.net.*;
import java.io.*;
import java.util.*;
public class Server
{
public static void main(String args[]) throws Exception
{
ServerSocket server=new ServerSocket(6262);
System.out.println("Server established.");
Socket client=server.accept();
ObjectOutputStream oos=new ObjectOutputStream(client.getOutputStream());
ObjectInputStream ois=new ObjectInputStream(client.getInputStream());
System.out.println("Client is now connected.");
int x=(Integer)ois.readObject();
int k=(Integer)ois.readObject();
int j=0;
int i=(Integer)ois.readObject();
boolean flag=true;
Random r=new Random(6);
int mod=r.nextInt(6);
while(mod==1||mod==0)
mod=r.nextInt(6);
while(true)
{
int c=k;
for(int h=0;h<=x;h++)
{
System.out.print("|"+c+"|");
c=(c+1)%x;
}
System.out.println();
System.out.println();
```

```
if(k==j)
{
System.out.println(“Frame “+k+” recieved”+”\n”+”Data:”+j);
j++;
System.out.println();
}

else
System.out.println(“Frames recieved not in correct order”+”\n”+” Expected farme:” + j
+”\n”+ ” Recieved frame no :”+ k);
System.out.println();
if(j%mod==0 && flag)
{
System.out.println(“Error found. Acknowledgement not sent. “);
flag=!flag;
j–;
}
else if(k==j-1)
{
oos.writeObject(k);
System.out.println(“Acknowledgement sent”);
}
System.out.println();
if(j%mod==0)
flag=!flag;
k=(Integer)ois.readObject();
if(k==-1)
break;
i=(Integer)ois.readObject();
}
System.out.println(“Client finished sending data. Exiting”);
oos.writeObject(-1);
}
}
```

/***Client Program***/

```
import java.util.*;
import java.net.*;
import java.io.*;
public class Client
{
public static void main(String args[]) throws Exception
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

```
System.out.print("Enter the value of m : ");
int m=Integer.parseInt(br.readLine());
int x=(int)((Math.pow(2,m))-1);
System.out.print("Enter no. of frames to be sent:");
int count=Integer.parseInt(br.readLine());
int data[]=new int[count];
int h=0;
for(int i=0;i<count;i++)
{
System.out.print("Enter data for frame no " +h+ " => ");
data[i]=Integer.parseInt(br.readLine());
h=(h+1)%x;
}
Socket client=new Socket("localhost",6262);
ObjectInputStream ois=new ObjectInputStream(client.getInputStream());
ObjectOutputStream oos=new ObjectOutputStream(client.getOutputStream());
System.out.println("Connected with server.");
boolean flag=false;
GoBackNListener listener=new GoBackNListener(ois,x);
listener=new GoBackNListener(ois,x);
listener.t.start();
int strt=0;
h=0;
oos.writeObject(x);
do
{
int c=h;
for(int i=h;i<count;i++)
{
System.out.print("|"+c+"|");
c=(c+1)%x;
}
System.out.println();
System.out.println();
h=strt;
for(int i=strt;i<x;i++)
{
System.out.println("Sending frame:"+h);
h=(h+1)%x;
System.out.println();
oos.writeObject(i);
oos.writeObject(data[i]);
Thread.sleep(100);
}
listener.t.join(3500);
```

```
if(listener.reply!=x-1)
{
System.out.println("No reply from server in 3.5 seconds. Resending data from frame no " +
(listener.reply+1));
System.out.println();
strt=listener.reply+1;
flag=false;
}
else
{
System.out.println("All elements sent successfully. Exiting");
flag=true;
}
}while(!flag);
oos.writeObject(-1);
}
}

class GoBackNListener implements Runnable
{
Thread t;
ObjectInputStream ois;
int reply,x;
GoBackNListener(ObjectInputStream o,int i)
{
t=new Thread(this);
ois=o;
reply=-2;
x=i;
}

public void run() {
try
{
int temp=0;
while(reply!=-1)
{
reply=(Integer)ois.readObject();
if(reply!=-1 && reply!=temp+1)
reply=temp;
if(reply!=-1)
{
temp=reply;
System.out.println("Acknowledgement of frame no " + (reply%x) + " recieved.");
System.out.println();
```

```
}
}
reply=temp;
}
catch(Exception e)
{
System.out.println(“Exception => ” + e);
}
}
}
```

Output:

/***Client Output**

```
Enter the value of m : 7
Enter no. of frames to be sent:5
Enter data for frame no 0 => 1
Enter data for frame no 1 => 2
Enter data for frame no 2 => 3
Enter data for frame no 3 => 4
Enter data for frame no 4 => 5
Connected with server.
|0||1||2||3||4|

Sending frame:0

Acknowledgement of frame no 0 recieved.

Sending frame:1

Sending frame:2

Sending frame:3

Sending frame:4

Sending frame:5
*/

/*Server Output

Server established.
Client is now connected.
|0||1||2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||
30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56
||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||8
3||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||1
07||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||
0|
```

Frame 0 recieved
Data:0
Acknowledgement sent

|1||2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1|

Frame 1 recieved
Data:1
Error found. Acknowledgement not sent.

|2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :2
|3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2||3|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :3
|4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2||3||4|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :4
*/

5.DISTANCE VECTOR ROUTING

1. **Aim**: Write a Program for DISTANCE VECTOR ROUTING
   Procedure:

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
   - It receives a distance vector from a neighbor containing different information than before.
   - It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

```
Dx(y) = Estimate of least cost from x to y
C(x,v) =  Node x knows cost to each neighbor v
Dx   =  [Dx(y): y ∈ N ] = Node x maintains distance vector
Node x also maintains its neighbors' distance vectors
- For each neighbor v, x maintains Dv = [Dv(y): y ∈ N ]
```

4. From time-to-time, each node sends its own distance vector estimate to neighbors.
5. When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

6. `Dx(y) = min { C(x,v) + Dv(y)} for each node y ∈ N`

Program:

```
import java.io.*;
public class DVR
{
 static int graph[][];
 static int via[][];
 static int rt[][];
 static int v;
 static int e;

 public static void main(String args[]) throws IOException
 {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

  System.out.println("Please enter the number of Vertices: ");
  v = Integer.parseInt(br.readLine());

  System.out.println("Please enter the number of Edges: ");
  e = Integer.parseInt(br.readLine());

  graph = new int[v][v];
  via = new int[v][v];
  rt = new int[v][v];
  for(int i = 0; i < v; i++)
```

```
  for(int j = 0; j < v; j++)
  {
   if(i == j)
    graph[i][j] = 0;
   else
    graph[i][j] = 9999;
  }

 for(int i = 0; i < e; i++)
 {
  System.out.println("Please enter data for Edge " + (i + 1) + ":");
  System.out.print("Source: ");
  int s = Integer.parseInt(br.readLine());
  s--;
  System.out.print("Destination: ");
  int d = Integer.parseInt(br.readLine());
  d--;
  System.out.print("Cost: ");
  int c = Integer.parseInt(br.readLine());
  graph[s][d] = c;
  graph[d][s] = c;
 }

 dvr_calc_disp("The initial Routing Tables are: ");

 System.out.print("Please enter the Source Node for the edge whose cost has changed: ");
 int s = Integer.parseInt(br.readLine());
 s--;
 System.out.print("Please enter the Destination Node for the edge whose cost has changed:
");
 int d = Integer.parseInt(br.readLine());
 d--;
 System.out.print("Please enter the new cost: ");
 int c = Integer.parseInt(br.readLine());
 graph[s][d] = c;
 graph[d][s] = c;

 dvr_calc_disp("The new Routing Tables are: ");
}

static void dvr_calc_disp(String message)
{
 System.out.println();
 init_tables();
 update_tables();
 System.out.println(message);
 print_tables();
 System.out.println();
}

static void update_table(int source)
{
```

```
 for(int i = 0; i < v; i++)
 {
  if(graph[source][i] != 9999)
  {
   int dist = graph[source][i];
   for(int j = 0; j < v; j++)
   {
    int inter_dist = rt[i][j];
    if(via[i][j] == source)
     inter_dist = 9999;
    if(dist + inter_dist < rt[source][j])
    {
     rt[source][j] = dist + inter_dist;
     via[source][j] = i;
    }
   }
  }
 }
}

static void update_tables()
{
 int k = 0;
 for(int i = 0; i < 4*v; i++)
 {
  update_table(k);
  k++;
  if(k == v)
   k = 0;
 }
}

static void init_tables()
{
 for(int i = 0; i < v; i++)
 {
  for(int j = 0; j < v; j++)
  {
   if(i == j)
   {
    rt[i][j] = 0;
    via[i][j] = i;
   }
   else
   {
    rt[i][j] = 9999;
    via[i][j] = 100;
   }
  }
 }
}
```

```
 static void print_tables()
 {
  for(int i = 0; i < v; i++)
  {
   for(int j = 0; j < v; j++)
   {
    System.out.print("Dist: " + rt[i][j] + "    ");
   }
   System.out.println();
  }
 }

}
```

output:-

```
Please enter the number of Vertices:
4
Please enter the number of Edges:
5
Please enter data for Edge 1:
Source: 1
Destination: 2
Cost: 1
Please enter data for Edge 2:
Source: 1
Destination: 3
Cost: 3
Please enter data for Edge 3:
Source: 2
Destination: 3
Cost: 1
Please enter data for Edge 4:
Source: 2
Destination: 4
Cost: 1
Please enter data for Edge 5:
Source: 3
Destination: 4
Cost: 4

The initial Routing Tables are:
Dist: 0    Dist: 1    Dist: 2    Dist: 2
Dist: 1    Dist: 0    Dist: 1    Dist: 1
Dist: 2    Dist: 1    Dist: 0    Dist: 2
Dist: 2    Dist: 1    Dist: 2    Dist: 0

Please enter the Source Node for the edge whose cost has changed: 2
Please enter the Destination Node for the edge whose cost has changed: 4
Please enter the new cost: 10

The new Routing Tables are:
Dist: 0    Dist: 1    Dist: 2    Dist: 6
Dist: 1    Dist: 0    Dist: 1    Dist: 5
Dist: 2    Dist: 1    Dist: 0    Dist: 4
Dist: 6    Dist: 5    Dist: 4    Dist: 0
```

6. CRC Code

Aim: Write a Program for CRC code

Procedure:

1. Start
2. Take a polynomial
3. Divide this with another data
4. It gives the remainder
5. We add this remainder to original polynomial
6. Sends to receiver
7. Receiver also calculates the crc16
8. If the remainder is all zeros
9. Then accepts the data
10. Otherwise rejects

Program:

```
public class CRC16 {

    /** value contains the currently computed CRC, set it to 0 initally */
    public int value;

    public CRC16() {
        value = 0;
    }

    /** update CRC with byte b */
    public void update(byte aByte) {
        int a, b;

        a = (int) aByte;
        for (int count = 7; count >=0; count--) {
            a = a << 1;
            b = (a >>> 8) & 1;
            if ((value & 0x8000) != 0) {
                value = ((value << 1) + b) ^ 0x1021;
            } else {
                value = (value << 1) + b;
            }
        }
        value = value & 0xffff;
        return;
    }

    /** reset CRC value to 0 */
    public void reset() {
        value = 0;
    }
}
```

Output:

x16 + x12 + x5 + 1

remainder: x^5+1

accept

7.Dijkstra's algorithm

Aim : Write a program for Dijkstra's algorithm

Procedure:

1. Start
2. Take a class dijksta
3. Take a function scanner to give input in that class
4. Read value into the matrix
5. calculate the distance matrix
6. Display the results
7. stop

Program:

```
import java.util.Scanner; //Scanner Function to take in the Input Values

public class Dijkstra
{
   static Scanner scan; // scan is a Scanner Object

   public static void main(String[] args)
   {
      int[] preD = new int[5];
      int min = 999, nextNode = 0; // min holds the minimum value, nextNode holds
the value for the next node.
      scan = new Scanner(System.in);
      int[] distance = new int[5]; // the distance matrix
      int[][] matrix = new int[5][5]; // the actual matrix
      int[] visited = new int[5]; // the visited array

      System.out.println("Enter the cost matrix");

      for (int i = 0; i < distance.length; i++)
      {
         visited[i] = 0; //initialize visited array to zeros
         preD[i] = 0;

         for (int j = 0; j < distance.length; j++)
         {
            matrix[i][j] = scan.nextInt(); //fill the matrix
            if (matrix[i][j]==0)
               matrix[i][j] = 999; // make the zeros as 999
         }
      }

      distance = matrix[0]; //initialize the distance array
      visited[0] = 1; //set the source node as visited
```

```
        distance[0] = 0; //set the distance from source to source to zero which is the
starting point

        for (int counter = 0; counter < 5; counter++)
        {
           min = 999;
           for (int i = 0; i < 5; i++)
           {
              if (min > distance[i] && visited[i]!=1)
              {
                 min = distance[i];
                 nextNode = i;
              }
           }

           visited[nextNode] = 1;
           for (int i = 0; i < 5; i++)
           {
              if (visited[i]!=1)
              {
                 if (min+matrix[nextNode][i] < distance[i])
                 {
                    distance[i] = min+matrix[nextNode][i];
                    preD[i] = nextNode;
                 }

              }

           }

        }

        for(int i = 0; i < 5; i++)
           System.out.print("|" + distance[i]);

        System.out.println("|");

        int j;
        for (int i = 0; i < 5; i++)
        {
           if (i!=0)
           {

              System.out.print("Path = " + i);
              j = i;
```

```
                do
                {
                    j = preD[j];
                    System.out.print(" <- " + j);
                }
                while(j != 0);
            }
            System.out.println();
        }
    }
}
```