Program (Bellman-ford Algorithm)

```
import java.util.*;
public class dva
{
   public static void main(String args[])
   {
      int i,j,k,m,n,x,y;
      int a[][]=new int[10][10];
      int h[][]=new int[10][10];
      Scanner s=new Scanner(System.in);
      System.out.println("Enter the number of nodes");
      n=s.nextInt();
      System.out.println("enter the distance matrix");
      for(i=1;i<=n;i++)
      {
         for(j=1;j<=n;j++)
         {
            a[i][j]=s.nextInt();
            h[i][j]=0;
         }
      }
      for(i=1;i<=n;i++)
      {
         for(j=1;j<=n;j++)
         {
            if(a[i][j]!=0 && a[i][j]!=999)
            {
               h[i][j]=1;
            }
```

```
    }
 }
System.out.println("the results before the calculations");
for(i=1;i<=n;i++)
{
   System.out.println("The routing table for "+i+" node number is");
   System.out.println("\n node \t distance \t hops");
   for(j=1;j<=n;j++)
   {
      System.out.println("--------------------------------------");
      System.out.println(j+"\t"+a[i][j]+"\t"+h[i][j]);
   }
}
for(m=1;m<=n;m++)
{
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      {
         int min=a[i][j];
         for(k=1;k<=n;k++)
         {
            if(min>a[i][k]+a[k][j])
            {
               a[i][j]=a[i][k]+a[k][j];
               h[i][j]=h[i][k]+h[k][j];
            }
         }
```

```
            }
          }
      }
      System.out.println("the results after the calculations");
      for(i=1;i<=n;i++)
      {
         System.out.println("The routing table for "+i+" node number is");
         System.out.println("\n node \t distance \t hops");
         for(j=1;j<=n;j++)
         {
            System.out.println("-------------------------------------");
            System.out.println(j+"\t"+a[i][j]+"\t"+h[i][j]);
         }
      }
      System.out.println("enter the nodes in between you want to find out the shortest
path");
      x=s.nextInt();
      y=s.nextInt();
      System.out.println("the shortest distance is "+a[x][y]+" and the hop value is "+h[x][y]);


   }
}
```

Program (leaky Bucket Algorithm)

```
import java.util.*;
public class leaky
{
   public static void main(String args[])
   {
      int store=0,incoming,outgoing,bucketsize,n;
      Scanner s=new Scanner(System.in);
      System.out.println("enter the bucket size ");
      bucketsize=s.nextInt();
      System.out.println("enter the incoming rate of water");
      incoming=s.nextInt();
      System.out.println("enter the outgoing rate of water");
      outgoing=s.nextInt();
      System.out.println("enter the number of inputs");
      n=s.nextInt();
      while(n!=0)
      {
         if(incoming<=bucketsize-store)
         {
            store=store+incoming;
            System.out.println("amount of water stored in the bucket "+store+" out of
"+bucketsize);
         }
         else
         {
            System.out.println("packetloss is "+(incoming-(bucketsize-store)));
            store=bucketsize;
```

```
            System.out.println("amount of water stored in the bucket "+store+" out of
"+bucketsize);
        }
        store=store-outgoing;
         System.out.println("after outgoing: "+store+" packet left out of "+bucketsize);
        n--;
     }
   }
}
```

Program (TCP Client and TCP Server)

TCP client

```
import java.io.*;
import java.net.*;
public class tcpclient
{
   public static void main(String args[]) throws Exception
   {
      Socket s=new Socket("localhost",4000);
      System.out.println("enter the file name");
      DataInputStream in=new DataInputStream(System.in);
      String fname=in.readLine();
      OutputStream os=s.getOutputStream();
      PrintWriter pw=new PrintWriter(os,true);
      pw.println(fname);
      InputStream is =s.getInputStream();
      DataInputStream read=new DataInputStream(is);
      String str;
      while((str=read.readLine())!=null)
      {
         System.out.println(str);
      }
      s.close();

   }
}
```

Tcp server

```
import java.io.*;
import java.net.*;
public class tcpserver
{
   public static void main(String args[]) throws Exception
   {
      ServerSocket ss=new ServerSocket(4000);
      System.out.println("server ready for connection");
      Socket s=ss.accept();
      System.out.println("waiting for the file name");
      InputStream is=s.getInputStream();
      DataInputStream in=new DataInputStream(is);
      String fname=in.readLine();
      BufferedReader read=new BufferedReader(new FileReader(fname));
      OutputStream os=s.getOutputStream();
      PrintWriter pw=new PrintWriter(os,true);
      String str;
      while((str=read.readLine())!=null)
      {
         pw.println(str);
      }
      s.close();
      ss.close();
   }
}
```

Program UDP

UDP Client

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class UDPClient {
public static void main(String[] args) throws Exception
{
Scanner sc=new Scanner(System.in);
InetAddress IP = InetAddress.getByName("localhost");
DatagramSocket clientSocket = new DatagramSocket();
while(true)
{
byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
System.out.println("Client: ");
String clientdata = sc.nextLine();
senddata = clientdata.getBytes();
DatagramPacket sendpacket = new DatagramPacket(senddata, senddata.length, IP, 9876);
clientSocket.send(sendpacket);
DatagramPacket receivePacket =new DatagramPacket(receivedata, receivedata.length);
clientSocket.receive(receivePacket);
String serverdata = new String(receivePacket.getData());
System.out.println("Server: " + serverdata);
}
}
}
```

UDP Server

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class UDPServer {
public static void main(String[] args) throws Exception
{
DatagramSocket serverSocket = new DatagramSocket(9876);
Scanner s=new Scanner(System.in);
while(true)
{
byte[] receivedata = new byte[1024];
byte[] senddata = new byte[1024];
DatagramPacket receivePacket = new DatagramPacket(receivedata, receivedata.length);
serverSocket.receive(receivePacket);
InetAddress IP = receivePacket.getAddress();
int port = receivePacket.getPort();
String clientdata = new String(receivePacket.getData());
System.out.println("Client : "+ clientdata);
System.out.println("Server : ");
String serverdata = s.nextLine();
senddata = serverdata.getBytes();
DatagramPacket sendPacket = new DatagramPacket(senddata, senddata.length, IP,port);
serverSocket.send(sendPacket);
}
}
}
```

Program (RSA)

```
import java.math.*;
import java.util.*;
import java.io.*;
 class RSA
{
BigInteger p, q, N, phi, e, d;
int bitlength = 100;
Random r;
RSA()
{
   r = new Random();
   p = BigInteger.probablePrime(bitlength, r);
   q = BigInteger.probablePrime(bitlength, r);
   N = p.multiply(q);
   phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
   e = BigInteger.probablePrime(bitlength/2, r);
   d = e.modInverse(phi);
}
public static void main(String[] args) throws Exception
{
RSA rsa = new RSA();
DataInputStream in=new DataInputStream(System.in);
String teststring ;
System.out.println("Enter the plain text:");
teststring=in.readLine();
System.out.println("Encrypting String: " + teststring);
System.out.println("String in Bytes: " + bytesToString(teststring.getBytes()));
byte[] encrypted = rsa.encrypt(teststring.getBytes());
```

```
System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted));
byte[] decrypted = rsa.decrypt(encrypted);
System.out.println("Decrypted String in Bytes: " + bytesToString(decrypted));
System.out.println("Decrypted String: ");
System.out.println(new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
   String test = "";
   for (byte b : encrypted)
  {
    test += Byte.toString(b);
  }
   return test;
}
public byte[] encrypt(byte[] message)
{
 return (new BigInteger(message)).modPow(e, N).toByteArray();
}
public byte[] decrypt(byte[] message)
{
 return (new BigInteger(message)).modPow(d, N).toByteArray();
}
}
```

Program (CRC)

```
import java.util.Scanner;
public class CRC
{
public static void main(String [] args)
{
Scanner s=new Scanner(System.in);
int databits,divisorbits,totlength,i;
System.out.print("Enter no of databits :");
databits=s.nextInt();
int data[]=new int[databits];
System.out.print("Enter databits :");
for(i=0;i<databits;i++)
data[i]=s.nextInt();
System.out.print("\nEnter no of divisor bits :");
divisorbits=s.nextInt();
int divisor[]=new int[divisorbits];
System.out.print("Enter divisorbits :");
for(i=0;i<divisorbits;i++)
divisor[i]=s.nextInt();
totlength=databits+divisorbits-1;
int div[]=new int[totlength];
int rem[]=new int[totlength];
int crc[]=new int[totlength];
for(i=0;i<data.length;i++)
div[i]=data[i];
System.out.print("CRC code after appending 0's : ");
for(i=0;i<div.length;i++)
{
```

```
System.out.print(div[i]);
rem[i]=div[i];
}
rem=divide(divisor,rem);
for(i=0;i<div.length;i++)
crc[i]=div[i]^rem[i];
System.out.print("\n\nCRC code is : ");
for(i=0;i<crc.length;i++)
System.out.print(crc[i]);
System.out.print("\nEnter the CRC code of "+totlength+" bits : ");
for(i=0;i<crc.length;i++)
{
crc[i]=s.nextInt();
rem[i]=crc[i];
}
rem=divide(divisor,rem);
for(i=0;i<rem.length;i++)
{
if(rem[i]!=0)
{
System.out.println("Error.");
break;
}
if(i==rem.length-1)
System.out.println("No Error.");
}
}
public static int[] divide(int divisor[],int rem[])
{
```

```
int cur=0,i;
while(true)
{
for(i=0;i<divisor.length;i++)
rem[cur+i]=rem[cur+i]^divisor[i];
while(rem[cur]==0 && cur!=rem.length-1) cur++;
if((rem.length-cur)<divisor.length) break;
}
return rem;
}
}
```