Pro7

```
import java.util.*;
class PRO7
{ void div(int a[],int k)
    { int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1}; //generating polynomial =
X^16 + x^12 + x^5 + 1 int
       count=0; for(int
       i=0;i<k;i++)
       { if(a[i]==gp[0])
           { for(int j=i;j<17+i;j++)
               { a[j]=a[j]^gp[count++];
               }
               count=0;
               }}}
   public static void main(String args[])
   { int a[]=new int[50];
       int b[]=new
       int[50]; int len,k;
       PRO7 ob=new PRO7(); //creating an object of class PRO7
       System.out.println("Enter the length of Data Frame:");
       Scanner scan=new Scanner(System.in); //Creating an object to invoke
Scanner Function to read objects len=scan.nextInt(); intflag=0:
       System.out.println("Enter the Message:");
       for(int i=0;i<len;i++)
       { a[i]=scan.nextInt();
       }

       for(int i=0;i<16;i++)
       { a[len++]=0;
       } k=len-16;

       for(int i=0;i<len;i++).
       { b[i]=a[i];
       }
       ob.div(a,k);
       for(int i=0;i<len;i++)
       a[i]=a[i]^b[i]; //produces data transmion bits
       System.out.println("Data to be transmitted: ");
       for(int i=0;i<len;i++)
       {
           System.out.print(a[i]+" ");
       }
       System.out.println();
       System.out.println("Enter the Reveived Data: ");
       for(int i=0;i<len;i++)
       { a[i]=scan.nextInt();
       }
       ob.div(a, k); //checkes with CRC-CCITT 16 bit. "Note not compare "
       for(int i=0;i<len;i++)
       { if(a[i]!=0)
           { flag=1;
              break;
           }
       } if(flag==1) //prints weather received data is correct or
       not.
       System.out.println("error in data");
       else
       System.out.println("no error");
   }
}
```

Pro 8

```
import java.util.Scanner;
class BELLMANFORD { static int n, dest;
static double[] prevDistanceVector, distanceVector;
static double[][] adjacencyMatrix;
public static void main(String[] args) { Scanner scanner = new
Scanner(System.in);
System.out.println("Enter number of nodes");
 n = scanner.nextInt();
adjacencyMatrix = new double[n][n];
 System.out.println("Enter Adjacency Matrix (Use 'Infinity' for No Link)");
for (int i = 0; i < n; i++)
for (int j = 0; j < n; j++)
adjacencyMatrix[i][j] = scanner.nextDouble();
System.out.println("Enter destination vertex");
dest = scanner.nextInt();
 distanceVector = new double[n];
for (int i = 0; i < n; i++)
distanceVector[i] = Double.POSITIVE_INFINITY;
distanceVector[dest - 1] = 0;
bellmanFordAlgorithm(); System.out.println("Distance Vector");
for (int i = 0; i < n; i++) { if (i == dest - 1)
{
continue;
}
System.out.println("Distance from " + (i + 1) + " is " + distanceVector[i]);
}
System.out.println();
}
static void bellmanFordAlgorithm()
{ for (int i = 0; i < n - 1; i++)
{ prevDistanceVector = distanceVector.clone();
 for (int j = 0; j < n; j++)
{ double min = Double.POSITIVE_INFINITY;
for (int k = 0; k < n; k++)
{
if (min > adjacencyMatrix[j][k] + prevDistanceVector[k])
{
min = adjacencyMatrix[j][k] + prevDistanceVector[k];
}
}
distanceVector[j] = min;
}}}}
```

Pro9

1

```
import java.net.*;
import java.io.*;
public class TCPClient
{
public static void main( String args[ ] ) throws Exception
{
      Socket sock = new Socket( "127.0.0.1", 4000);
      System.out.print("Enter the file name\n");
      BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
      String fname = br.readLine();
      OutputStream ostream = sock.getOutputStream( );
      PrintWriter pwrite = new PrintWriter(ostream, true);
      pwrite.println(fname);

      InputStream istream = sock.getInputStream();
      BufferedReader socketRead = new BufferedReader(new
      InputStreamReader(istream));
      String str;

      while((str = socketRead.readLine()) != null)
      {
            System.out.println(str);
      }

      pwrite.close(); socketRead.close(); br.close(); sock.close();
}
}
```

2

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
public class TCPServer
{
public static void main(String args[]) throws Exception
{
      ServerSocket sersock = new ServerSocket(4000);
      System.out.println("Server ready for connection");
      Socket sock = sersock.accept();
      System.out.println("Connection successful | wating for filename");
      InputStream istream = sock.getInputStream( );
      BufferedReader br =new BufferedReader(new InputStreamReader(istream));
      String fname = br.readLine( );
      BufferedReader contentRead = new BufferedReader(new FileReader(fname) );
      OutputStream ostream = sock.getOutputStream( );
      PrintWriter pwrite = new PrintWriter(ostream, true);
      String str;
      while((str = contentRead.readLine()) != null)
      {
            pwrite.println(str);
      }
      System.out.println("File Contents sent successfully");
      sock.close(); sersock.close();
      pwrite.close(); br.close(); contentRead.close();
}
}
```

Pro10
1

```
import java.io.*;
import java.net.*;
class UDPClient
{
public static DatagramSocket clientsocket;
public static DatagramPacket dp;
public static BufferedReader br;
public static InetAddress ia;
public static byte buf[] = new byte[1024];
public static int cport = 222, sport = 555;
public static void main(String[] args) throws IOException
{
      clientsocket = new DatagramSocket(cport);
      dp = new DatagramPacket(buf, buf.length);
      br = new BufferedReader(new InputStreamReader(System.in));
      ia = InetAddress.getLocalHost();

      System.out.println("Client is Running...");
      System.out.println("Type some text if u want to Quit type 'exit'.");
      while(true)
      {
            String str1 = new String(br.readLine());
            buf = str1.getBytes();
            if(str1.equals("exit"))
            {
                  System.out.println("Terminated..");
                  clientsocket.send(new
DatagramPacket(buf,str1.length(),ia,sport));
                  break;
            }
            clientsocket.send(new DatagramPacket(buf,str1.length(), ia,
sport));

            clientsocket.receive(dp);
            String str4 = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Server said : " + str4);
      }
}
}
```

2

Same as client(change client to server)till System.out.println("Client is Running...");

```
         System.out.println("Server is Running...");
         while(true)
         {
          serversocket.receive(dp);
          String str2 = new String(dp.getData(), 0, dp.getLength());
          if(str2.equals("exit"))
          {
           System.out.println("Terminated...");
           break;
          }
          System.out.println("Client said : " + str2);

          String str3 = new String(br.readLine());
          buf = str3.getBytes();
          serversocket.send(new DatagramPacket(buf,str3.length(), ia, cport));
         }
```

```
        }
    }
```

Pro11

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

class PRO11 {

    static BigInteger p, q, n, phi_n, e, d;
    static SecureRandom secureRandom;
    static int bitLength = 64;

    static String encrypt(String msg) {
        return new BigInteger(msg.getBytes()).modPow(e, n).toString();
    }

    static String decrypt(String cipher) {
        BigInteger bi = new BigInteger(cipher).modPow(d, n);
        return new String(bi.toByteArray());
    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        secureRandom = new SecureRandom();

        p = BigInteger.probablePrime(bitLength, secureRandom);
        q = BigInteger.probablePrime(bitLength, secureRandom);
        n = p.multiply(q);
        phi_n =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

        e = BigInteger.probablePrime(bitLength / 2, secureRandom);
        while (e.gcd(phi_n).compareTo(BigInteger.ONE) != 0 &&
e.compareTo(phi_n) < 0) {
            e = e.add(BigInteger.ONE);
        }

        d = e.modInverse(phi_n);

        System.out.println("P assigned as: " + p);
        System.out.println("Q assigned as: " + q);
        System.out.println("N assigned as: " + n);
        System.out.println("PHI_N assigned as: " + phi_n);

        System.out.println("\nEnter Message");
        String msg = scanner.nextLine();

        String encryptedMessage = encrypt(msg);
        System.out.println("Encrypted Message: " + encryptedMessage);

        String decryptedMessage = decrypt(encryptedMessage);
        System.out.println("Decrypted Message: " + decryptedMessage);

    }
}
```

Pro12

```
import java.io.*;
import java.util.*;

class PRO12 {
private static int
no_of_packet,bucket_capacity,array_size,current_bucket,over_flow,fixed_data_fl
ow;
private static int array[];
public void LeakyBucket()
{
      current_bucket=0;
      System.out.print("\nCurrent Bucket size :" +current_bucket );
      for(int i=0;i<array_size;i++)
      {
            int input=array[i];
            System.out.print("\n----------------------\nInput to bucket is "
+input );
            over_flow=0;
            current_bucket=current_bucket+input;
            System.out.print("\nCurrent Bucket size :"+current_bucket);
            if(current_bucket<=fixed_data_flow)
                  current_bucket=0;
            else
            {
                  current_bucket=current_bucket-fixed_data_flow;
            }

            if(current_bucket<=bucket_capacity)
            {
                  System.out.print("\nNO OverFLOW");
            }
            else
            {
                  over_flow=(current_bucket-bucket_capacity);
                  current_bucket=bucket_capacity;
                  System.out.print("\nOver Flow Occured :"+over_flow);
            }
            System.out.print("\n:"+current_bucket);}}
            public static void main(String args[]){
      PRO12 pr = new PRO12();
      Scanner scan = new Scanner(System.in);
      System.out.print("Enter the Bucket Capacity : ");
      bucket_capacity = scan.nextInt();
      System.out.print("\nEnter the Bucket Fixed Data Flow : ");
      fixed_data_flow = scan.nextInt();
      System.out.print("\nEnter the Array Size : ");
      array_size=scan.nextInt();
      System.out.println("\nEnter the Input values of size : "+array_size);
      array = new int[array_size];
      for(int i=0;i<array_size;i++)
      {array[i]=scan.nextInt();}
      System.out.println("The Input for LeakyBucket is ");
      for(int i=0;i<array_size;i++)
      {
            System.out.print(array[i]+ " " );
      }
      pr.LeakyBucket();
      System.out.print("\n\nPROGRAM TERMINATING SUCCESSFULLY...\n");
      scan.close();}}
```