P7

```
import java.util.Scanner;
class CRC
{
 static String datastream;
static String generator= "10001000000100001";
public static void main(String args[])
{
 Scanner sc = new Scanner(System.in);
System.out.println("---At the Sender---\n Enter data stream: ");
String datastream = sc.nextLine();
int datalen=datastream.length();
int genlen =generator.length();

int data[] = new int[datalen + genlen - 1];
int codeword[] = new int[datalen + genlen - 1];
int div[] = new int[generator.length()];
for(int i=0;i<datastream.length();i++)
data[i] = Integer.parseInt(datastream.charAt(i)+"");
for(int i=0;i<generator.length();i++)
div[i] = Integer.parseInt(generator.charAt(i)+"");

codeword = calculateCrc(data,div,datalen);
System.out.println("The CRC(Final Codeword) code is: ");
//Display CRC- final codeword
for(int i=0;i<datastream.length();i++)
codeword[i] = Integer.parseInt(datastream.charAt(i)+"");
for(int i=0;i<data.length;i++)
System.out.print(codeword[i]);
System.out.println("\n");

System.out.println("---At the Receiver---\n Enter Received codeword: ");
//Check for input CRC code
datastream = sc.nextLine();
data = new int[datastream.length() + generator.length() - 1];
for(int i=0;i<datastream.length();i++)
data[i] = Integer.parseInt(datastream.charAt(i)+"");
codeword = calculateCrc(data,div,datalen);
boolean valid = true; //Checking remainder is zero or not
for(int i=0;i<codeword.length;i++)
if(codeword[i]==1)
{
valid = false;
break;
}

if(valid==true)
System.out.println("Data stream is valid. No error occured");
```

```
else
System.out.println("Data stream is invalid. CRC error occured.");
sc.close();
}

public static int[] calculateCrc(int[] divrem,int[] divisor,int len)
{ //Calculation of CRC
for(int i=0;i<len;i++)
{
if(divrem[i]==1)
for(int j=0;j<divisor.length;j++)
divrem[i+j] ^= divisor[j];
}
return divrem;
}
}
```

P8

```
import java.util.Scanner;
public class BellmanFord
{
private int dist[];
private int noofvert;
public static final int MAXVAL = 999;
public BellmanFord(int noofvert)
{
this.noofvert = noofvert;
dist = new int[noofvert + 1];
}

public void BellmanFordEval(int source, int adjmtx[][])
{
for (int node = 1; node <= noofvert; node++)
{
dist[node] = MAXVAL;
}
dist[source] = 0;
for (int node = 1; node <= noofvert - 1; node++)
{
for (int sn = 1; sn<= noofvert; sn++)
{
for (int dn = 1; dn<= noofvert; dn++)
{
if (adjmtx[sn][dn] != MAXVAL)
{
if (dist[dn] > dist[sn] + adjmtx[sn][dn])
dist[dn] = dist[sn] + adjmtx[sn][dn];
}
}
}
}

System.out.println("After Nth Iteration");
for (int v = 1; v <= noofvert; v++)
{
System.out.println("distance of source " + source + " to " + v + " is " +
dist[v]);
}
}

public static void main(String[] args)
{
int noofvert = 0;
```

```
int source;
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the number of vertices");
noofvert = scanner.nextInt();
int adjmtx[][] = new int[noofvert + 1][noofvert + 1];
System.out.println("Enter the adjacency matrix");

for (int sn = 1; sn<= noofvert; sn++)
{
for (int dn = 1; dn<= noofvert; dn++)
{
adjmtx[sn][dn] = scanner.nextInt();
 if (sn == dn)
{
adjmtx[sn][dn] = 0;
continue;
}
if (adjmtx[sn][dn] == 0)
{
adjmtx[sn][dn] = MAXVAL;
}
}
}

System.out.println("Enter the source vertex");
source = scanner.nextInt();
BellmanFord bellmanford = new BellmanFord(noofvert);
bellmanford.BellmanFordEval(source, adjmtx);
scanner.close();
}
}
```

P9

CLIENT SIDE PROGRAM

```
import java.io.*;
import java.net.*;
public class FileClient
{
public static void main(String[] args)
{
new FileClient();
}
//TCP Socket Program
public FileClient()
{
BufferedReader bufReader=new BufferedReader(new InputStreamReader(System.in));
try
{
System.out.println("Enter IP address of the server:");
String saddr = bufReader.readLine();
Socket clientsocket=new Socket(saddr,8000);
System.out.println("Connecting to Server....");
DataInputStream input=new DataInputStream(clientsocket.getInputStream());
DataOutputStream output=new DataOutputStream(clientsocket.getOutputStream());
System.out.println("Enter File Name:");
String Name=bufReader.readLine();
output.writeUTF(Name);
```

```
String EchoedFile=input.readUTF();

System.out.println("----------------------------------------------");

System.out.println("Content of a File:\n\n"+EchoedFile);

System.out.println("----------------------------------------------");

clientsocket.close();

}

//TCP Socket Program

catch(IOException ex)

{

ex.printStackTrace();

}

}

}
```

SERVER SIDE PROGRAM

```
import java.io.* ; import java.net.*;
public class FileServer
{
public static void main(String[] args)
{
new FileServer();
}
public FileServer()
{
try
{
ServerSocket serversocket=new ServerSocket(8000);
System.out.println("Server Started....");
System.out.println("----------------------------------------------");

Socket socket=serversocket.accept();
DataInputStream input=new DataInputStream(socket.getInputStream());
DataOutputStream output=new DataOutputStream(socket.getOutputStream());
String str=input.readUTF();
System.out.println("Requested File Name:"+str);
System.out.println("----------------------------------------------");
String everything="";
```

```
try
{
InputStream in = new FileInputStream(str);
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
StringBuilder out = new StringBuilder();
String line;
System.out.println("Reading Contents of the File...");
System.out.println("-----------------------------------------------");
while ((line = reader.readLine()) != null)
{ out.append(line+"\n");
}
everything= out.toString();
System.out.println("File Contents sent to client...");
System.out.println("-----------------------------------------------");
}
catch(Exception ex)
{
everything="File Not Found!";
}
output.writeUTF(everything);
}
catch(Exception ex)
{
ex.printStackTrace();
}
}
}
```

P10

CLIENT SIDE PROGRAM

```
import java.io.*;
import java.net.*;
 class UDPClient
{
public static void main(String args[]) throws Exception
{
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the IP address of the Server:");
String saddr = inFromUser.readLine();
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName(saddr);
byte[] receiveData;
byte[] sendData = new byte[200];
String sentence = "Hello";
sendData = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
9876);
clientSocket.send(sendPacket);
while(true)
{
receiveData = new byte[200];
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String incomingData = new String(receivePacket.getData());
InetAddress SAddress = receivePacket.getAddress();
System.out.println("FROM SERVER"+"("+SAddress.toString()+"): " + incomingData);
System.out.println("--------------------------------------------------------------");
}
}
}
```

SERVER SIDE PROGRAM

```
import java.io.*;
import java.net.*;
class UDPServer
{
public static void main(String args[]) throws Exception
{
```

```
DatagramSocket serverSocket = new DatagramSocket(9876);
System.out.println("--------Server Started-------");
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
byte[] receiveData = new byte[200];
byte[] sendData;
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
serverSocket.receive(receivePacket);
InetAddress clientAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
System.out.println("Client with IP Address "+clientAddress.toString()+"
connected...");
System.out.println("-----------------------------------------------------------");
System.out.println("Enter the message to send to client:");
while(true)
{
String input = inFromUser.readLine();
sendData = new byte[200];
sendData = input.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress, port);
serverSocket.send(sendPacket);
}
}
}
```

P11

```
import java.math.BigInteger;
import java.util.*;

class RSA
{
public static void main(String args[])
{
Scanner ip=new Scanner(System.in);
int p,q,n,e=1,j;
int d=1,i1; int t1,t2;
int pt[]= new int[10];
int ct[]= new int[10];
int rt[]= new int[10];
int temp[]= new int[10];
String i=new String();

System.out.println("Enter the two prime numbers:");
p=ip.nextInt();
q=ip.nextInt();

System.out.println("Enter the message to be sent");
i=ip.next();
i1=i.length(); n=p*q;
t1=p-1;
t2=q-1;
System.out.println("\n----------------------");
System.out.println("Sender Side:");
while((t1*t2)%e==0)
	e++;

System.out.println("Public Key(e)= "+e);
System.out.println("--------------------");
for(j=0;j<i1;j++)
{
pt[j]=(i.charAt(j))-96;
System.out.println("Plain Text= "+pt[j]);
ct[j]=((int)Math.pow(pt[j],e))%n;
System.out.println("Cipher Text= "+ct[j]);
}
System.out.println("\nTransmitted Message:"); for(j=0;j<i1;j++)
{
	temp[j]=ct[j]+96;
	System.out.print((char)temp[j]);
}
System.out.println("\n\n----------------------");
System.out.println("Receiver Side:");
while((d*e)%(t1*t2)!=1)
	d++;

System.out.println("Private Key(d)= "+d);
```

```
System.out.println("----------------------");
for(j=0;j<i1;j++)
{
System.out.println("cipher Text= "+ct[j]);
BigInteger very_big_no = BigInteger.valueOf(ct[j]);
very_big_no = very_big_no.pow(d);
very_big_no = very_big_no.mod(BigInteger.valueOf(n));
rt[j] = very_big_no.intValue();
System.out.println("Plain Text= "+rt[j]);
}
System.out.println("\n----------------------");
System.out.println("Decrypted Message:");
for(j=0;j<i1;j++)
{
      rt[j]=rt[j]+96; System.out.print((char)rt[j]);
}
System.out.println("\n----------------------");
ip.close();
}
}
```

P12

```
import java.util.Scanner;
public class LeakyBucket
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int incoming, outgoing, buck_size, n, time = 1, store = 0;
System.out.println("Enter bucket size, outgoing rate and Number of Packets:");
buck_size = sc.nextInt();
outgoing = sc.nextInt();
n = sc.nextInt();
while (n != 0) {
System.out.println("Enter the incoming packet size at Time:" + (time++) );
incoming=sc.nextInt();
System.out.println("Incoming packet size is " + incoming);
if (incoming <= (buck_size - store))
{
	store += incoming;
	System.out.println("Bucket buffer size is " + store + " out of " + buck_size);
}
else
{
	int pktdrop = incoming - (buck_size - store);
	System.out.println("Dropped " + pktdrop + " no of packets");
	System.out.println("Bucket buffer size is 10 out of "+ buck_size);
	store = buck_size;
}
store = store - outgoing;
if(store < 0)
{
	store=0;
	System.out.println("Empty Buffer");
}
System.out.println("After outgoing: "+ store +" packets left out of " + buck_size + "
in buffer\n");
n--;
}
sc.close();
}
}
```