CN Lab Programs

1.CRC?

```
import java.util.*;
class CRCExample {
    public static void main(String args[])
    {
        Scanner scan = new Scanner(System.in);

    int size;
        System.out.println("Enter the size of the data array: ");
    size = scan.nextInt();
        int data[] = new int[size];
        System.out.println("Enter data bits in the array one by one: ");
    for(int i = 0 ; i < size ; i++)
    {
      System.out.println("Enter bit " + (size-i) + ":");
      data[i] = scan.nextInt();
    }
    System.out.println("Enter the size of the divisor array:");
    size = scan.nextInt();

    int divisor[] = new int[size];
    System.out.println("Enter divisor bits in the array one by one: ");
    for(int i = 0 ; i < size ; i++)
    {
      System.out.println("Enter bit " + (size-i) + ":");
```

```
    divisor[i] = scan.nextInt();
}

int rem[] = divideDataWithDivisor(data, divisor);

for(int i = 0; i < rem.length-1; i++)
{
    System.out.print(rem[i]);
}
System.out.println("\nGenerated CRC code is: ");

for(int i = 0; i < data.length; i++)
{
    System.out.print(data[i]);
}
for(int i = 0; i < rem.length-1; i++)
{
    System.out.print(rem[i]);
}
System.out.println();

int sentData[] = new int[data.length + rem.length - 1];
System.out.println("Enter bits in the array which you want to send: ");
for(int i = 0; i < sentData.length; i++)
{
    System.out.println("Enter bit " +(sentData.length - 1)+ ":");
    sentData[i] = scan.nextInt();
```

```
    }
    receiveData(sentData, divisor);
}

static int[] divideDataWithDivisor(int oldData[], int divisor[])
{

    int rem[] = new int[divisor.length];
    int i;
    int data[] = new int[oldData.length + divisor.length];

    System.arraycopy(oldData, 0, data, 0, oldData.length);
    System.arraycopy(data, 0, rem, 0, divisor.length);

    for(i = 0; i < oldData.length; i++) {
        System.out.println((i+1) + ".) First data bit is : "+ rem[0]);
        System.out.print("Remainder : ");
        if(rem[0] == 1) {

            for(int j = 1; j < divisor.length; j++)
            {
                rem[j-1] = exorOperation(rem[j], divisor[j]);
                System.out.print(rem[j-1]);
            }
        }
        else {
```

```
            for(int j = 1; j < divisor.length; j++)
            {
                rem[j-1] = exorOperation(rem[j], 0);
                System.out.print(rem[j-1]);
            }
        }

        rem[divisor.length-1] = data[i+divisor.length];
        System.out.println(rem[divisor.length-1]);
    }
    return rem;
}
static int exorOperation(int x, int y)
{
    if(x == y) {
        return 0;
    }
    return 1;
}
static void receiveData(int data[], int divisor[])
{
    int rem[] = divideDataWithDivisor(data, divisor);

    for(int i = 0; i < rem.length; i++)
    {
        if(rem[i] != 0) {
```

```
            System.out.println("Currupted data received...");
            return;
        }
    }
    System.out.println("Data received without any error.");
  }
}
```

```
C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java CRC.java
Enter the size of the data array:
7
Enter data bits in the array one by one:
Enter bit 7:
1 0 0 1 1 0 1
Enter bit 6:
Enter bit 5:
Enter bit 4:
Enter bit 3:
Enter bit 2:
Enter bit 1:
Enter the size of the divisor array:
4
Enter divisor bits in the array one by one:
Enter bit 4:
1 0 1 1
Enter bit 3:
Enter bit 2:
Enter bit 1:
1.) First data bit is : 1
Remainder : 0101
2.) First data bit is : 0
Remainder : 1010
3.) First data bit is : 1
Remainder : 0011
4.) First data bit is : 0
Remainder : 0110
5.) First data bit is : 0
Remainder : 1100
6.) First data bit is : 1
Remainder : 1110
7.) First data bit is : 1
Remainder : 1010
101
Generated CRC code is:
1001101101
Enter bits in the array which you want to send:
Enter bit 9:
1 0 0 1 1 0 1 1 0 1 1 0 1
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
Enter bit 9:
```

```
1.) First data bit is : 1
Remainder : 0101
2.) First data bit is : 0
Remainder : 1010
3.) First data bit is : 1
Remainder : 0011
4.) First data bit is : 0
Remainder : 0111
5.) First data bit is : 0
Remainder : 1110
6.) First data bit is : 1
Remainder : 1011
7.) First data bit is : 1
Remainder : 0000
8.) First data bit is : 0
Remainder : 0000
9.) First data bit is : 0
Remainder : 0000
10.) First data bit is : 0
Remainder : 0000
Data received without any error.
```

1.CRC (VERSION - 2)

```
import java.util.Scanner;
import java.io.*;
public class CRC1
{
        public static void main(String args[])
        {
        Scanner sc = new Scanner(System.in);
        //Input Data Stream
        System.out.print("Enter message bits: ");
        String message = sc.nextLine();
        System.out.print("Enter generator: ");
        String generator = sc.nextLine();
    int data[] = new int[message.length() + generator.length() - 1];
    int divisor[] = new int[generator.length()];
    for(int i=0;i<message.length();i++)
            data[i] = Integer.parseInt(message.charAt(i)+"");
    for(int i=0;i<generator.length();i++)
            divisor[i] = Integer.parseInt(generator.charAt(i)+"");
    //Calculation of CRC
```

```
for(int i=0;i<message.length();i++)
{
        if(data[i]==1)
                for(int j=0;j<divisor.length;j++)
                        data[i+j] ^= divisor[j];
}
//Display CRC
System.out.print("The checksum code is: ");
for(int i=0;i<message.length();i++)
        data[i] = Integer.parseInt(message.charAt(i)+"");
for(int i=0;i<data.length;i++)
   System.out.print(data[i]);
System.out.println();
//Check for input CRC code
System.out.print("Enter checksum code: ");
        message = sc.nextLine();
System.out.print("Enter generator: ");
        generator = sc.nextLine();
data = new int[message.length() + generator.length() - 1];
divisor = new int[generator.length()];
for(int i=0;i<message.length();i++)
        data[i] = Integer.parseInt(message.charAt(i)+"");
for(int i=0;i<generator.length();i++)
        divisor[i] = Integer.parseInt(generator.charAt(i)+"");
//Calculation of remainder
for(int i=0;i<message.length();i++)
{
```

```
            if(data[i]==1)
                for(int j=0;j<divisor.length;j++)
                    data[i+j] ^= divisor[j];
        }
        //Display validity of data
        boolean valid = true;
        for(int i=0;i<data.length;i++)
            if(data[i]==1){
                valid = false;
                break;
        }
        if(valid==true)
            System.out.println("Data stream is valid");
        else
            System.out.println("Data stream is invalid. CRC error occurred.");
        }
}
```

```
C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java CRC1.java
Enter message bits: 1101011011
Enter generator: 10011
The checksum code is: 11010110111110
Enter checksum code: 11010110111110
Enter generator: 10011
Data stream is valid

C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java CRC1.java
Enter message bits: 1101011011
Enter generator: 10011
The checksum code is: 11010110111110
Enter checksum code: 11010110110110
Enter generator: 10011
Data stream is invalid. CRC error occurred.

C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>
```

2.BellamFord

```
import java.util.Scanner;
public class BellmanFord
{
   private int distances[];
   private int numberofvertices;
   public static final int MAX_VALUE = 999;
   public BellmanFord(int numberofvertices)
   {
      this.numberofvertices = numberofvertices;
      distances = new int[numberofvertices + 1];
   }
   public void BellmanFordEvaluation(int source, int adjacencymatrix[][])
   {
      for (int node = 1; node <= numberofvertices; node++)
      {
         distances[node] = MAX_VALUE;
      }
      distances[source] = 0;
      for (int node = 1; node <= numberofvertices - 1; node++)
      {
         for (int sourcenode = 1; sourcenode <= numberofvertices; sourcenode++)
         {
            for (int destinationnode = 1; destinationnode <= numberofvertices;
destinationnode++)
            {
               if (adjacencymatrix[sourcenode][destinationnode] != MAX_VALUE)
```

```
            {
                if (distances[destinationnode] > distances[sourcenode]
                        + adjacencymatrix[sourcenode][destinationnode])
                    distances[destinationnode] = distances[sourcenode]
                        + adjacencymatrix[sourcenode][destinationnode];
            }
        }
    }
}
for (int sourcenode = 1; sourcenode <= numberofvertices; sourcenode++)
{
    for (int destinationnode = 1; destinationnode <= numberofvertices; destinationnode++)
    {
        if (adjacencymatrix[sourcenode][destinationnode] != MAX_VALUE)
        {
            if (distances[destinationnode] > distances[sourcenode]
                    + adjacencymatrix[sourcenode][destinationnode])
                System.out.println("The Graph contains negative egde cycle");
        }
    }
}
for (int vertex = 1; vertex <= numberofvertices; vertex++)
{
    System.out.println("distance of source  " + source + " to "
            + vertex + " is " + distances[vertex]);
}
}
```

```
public static void main(String... arg)
{
   int numberofvertices = 0;
   int source;
   Scanner scanner = new Scanner(System.in);
   System.out.println("Enter the number of vertices");
   numberofvertices = scanner.nextInt();
   int adjacencymatrix[][] = new int[numberofvertices + 1][numberofvertices + 1];
   System.out.println("Enter the adjacency matrix");
   for (int sourcenode = 1; sourcenode <= numberofvertices; sourcenode++)
   {
      for (int destinationnode = 1; destinationnode <= numberofvertices; destinationnode++)
      {
         adjacencymatrix[sourcenode][destinationnode] = scanner.nextInt();
            if (sourcenode == destinationnode)
         {
            adjacencymatrix[sourcenode][destinationnode] = 0;
            continue;
         }
         if (adjacencymatrix[sourcenode][destinationnode] == 0)
         {
            adjacencymatrix[sourcenode][destinationnode] = MAX_VALUE;
         }
      }
   }
   System.out.println("Enter the source vertex");
```

```
        source = scanner.nextInt();
        BellmanFord bellmanford = new BellmanFord(numberofvertices);
        bellmanford.BellmanFordEvaluation(source, adjacencymatrix);
        scanner.close();
    }
}
```

```
C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java BellmanFord.java
Enter the number of vertices
5
Enter the adjacency matrix
0 1 5 0 0
1 0 3 0 9
5 3 0 4 0
0 0 4 0 2
0 9 0 2 0
Enter the source vertex
2
distance of source  2 to 1 is 1
distance of source  2 to 2 is 0
distance of source  2 to 3 is 3
distance of source  2 to 4 is 7
distance of source  2 to 5 is 9
```

3.TCP

//TCPServer

```
import java.net.*;
import java.io.*;

public class TCPS
{
	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 Is successful and waiting for chatting");
		InputStream istream=sock.getInputStream();
		BufferedReader fileRead=new BufferedReader(new
InputStreamReader(istream));
		String fname=fileRead.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);
		}
		sock.close();
		sersock.close();
```

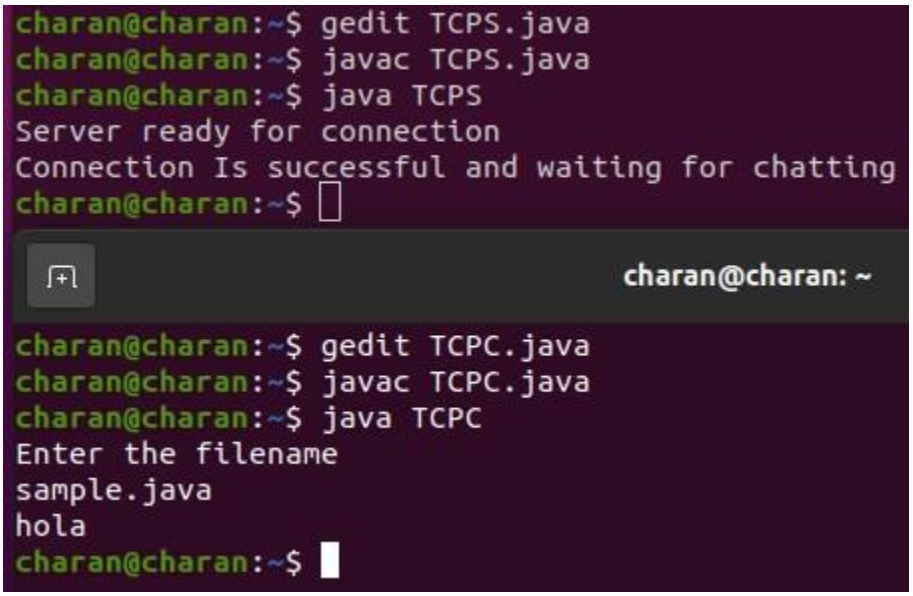
```
                pwrite.close();
                fileRead.close();
                ContentRead.close();
        }
}
```

//TCPClient

//create sample file(sample.java)

```
import java.net.*;
import java.io.*;
public class TCPC
{
        public static void main(String[] args) throws Exception
        {
                Socket sock=new Socket("127.0.01",4000);
                System.out.println("Enter the filename");
                BufferedReader keyRead=new BufferedReader(new
InputStreamReader(System.in));
                String fname=keyRead.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();
            keyRead.close();
        }
}
```



4.UDP

//UDPServer

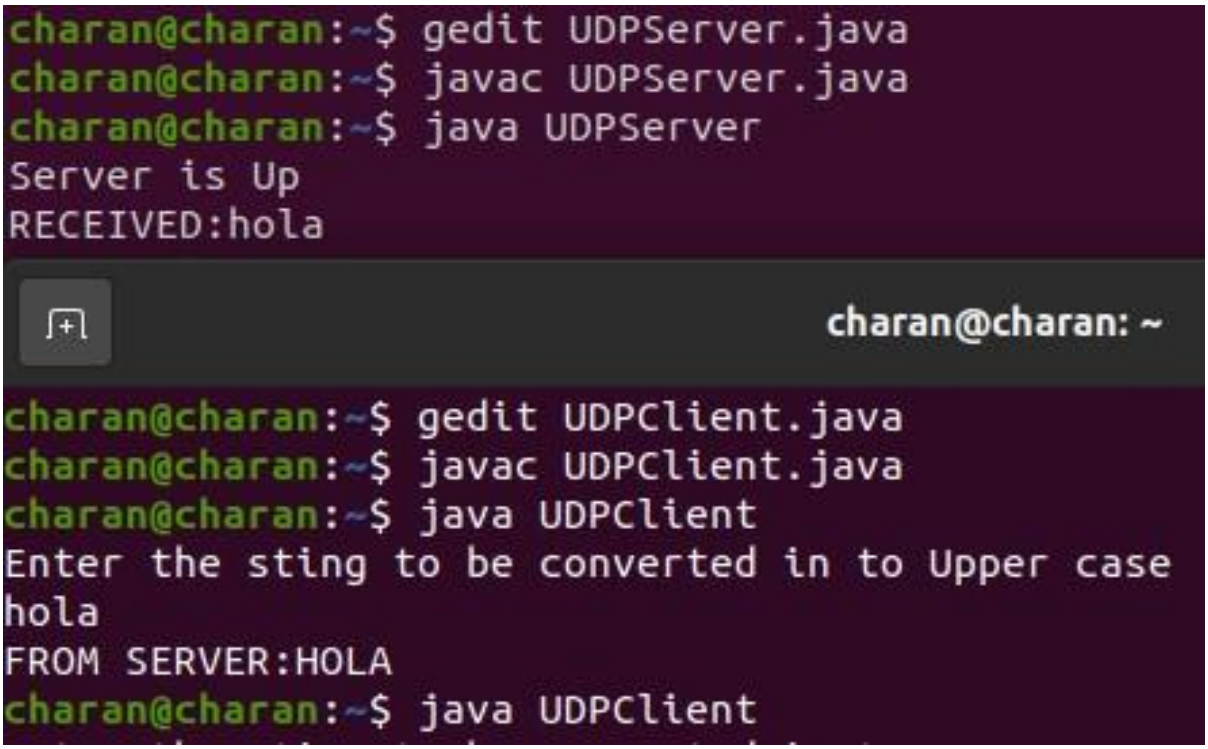
```
import java.net.*;
import java.net.InetAddress;

class UDPServer
{
	public static void main(String args[])throws Exception
	{
		DatagramSocket serverSocket = new DatagramSocket(9876);
		byte[] receiveData=new byte[1024];
		byte[] sendData=new byte[1024];
		while(true)
		{
			System.out.println("Server is Up");
			DatagramPacket receivePacket=new
DatagramPacket(receiveData,receiveData.length);
			serverSocket.receive(receivePacket);
			String sentence=new String(receivePacket.getData());
			System.out.println("RECEIVED:"+sentence);
			InetAddress IPAddress=receivePacket.getAddress();
			int port=receivePacket.getPort();
			String capitalizedSentence=sentence.toUpperCase();
			sendData=capitalizedSentence.getBytes();
			DatagramPacket sendPacket=new
```

```
                DatagramPacket(sendData,sendData.length,IPAddress,port);
                serverSocket.send(sendPacket);
            }
        }
}
```

//UDPClient

```
import java.io.*;
import java.net.*;
import java.net.InetAddress;

class UDPClient
{
        public static void main(String[] args)throws Exception
        {
            BufferedReader inFromUser=new BufferedReader(new
InputStreamReader(System.in));
            DatagramSocket clientSocket=new DatagramSocket();
            InetAddress IPAddress=InetAddress.getByName("localhost");
            byte[] sendData=new byte[1024];
            byte[] receiveData=new byte[1024];
            System.out.println("Enter the sting to be converted in to Upper case");
            String sentence=inFromUser.readLine();
            sendData=sentence.getBytes();
            DatagramPacket sendPacket=new
            DatagramPacket(sendData,sendData.length,IPAddress,9876);
            clientSocket.send(sendPacket);
```

```
            DatagramPacket receivePacket=new
DatagramPacket(receiveData,receiveData.length);

            clientSocket.receive(receivePacket);

            String modifiedSentence=new String(receivePacket.getData());

            System.out.println("FROM SERVER:"+modifiedSentence);

            clientSocket.close();

        }

}
```



5.RSA

```
import java.math.*;
class RSA {
    public static void main(String args[])
    {
        int p, q, n, z, d = 0, e, i;
        int msg = 12;
        double c;
        BigInteger msgback;
        p = 3;
        q = 11;
        n = p * q;
        z = (p - 1) * (q - 1);
        System.out.println("the value of z = " + z);

        for (e = 2; e < z; e++)
        {
            if (gcd(e, z) == 1)
            {
                break;
            }
        }
        System.out.println("the value of e = " + e);
        for (i = 0; i <= 9; i++)
        {
```

```
            int x = 1 + (i * z);
            if (x % e == 0)
            {
                d = x / e;
                break;
            }
        }
        System.out.println("the value of d = " + d);
        c = (Math.pow(msg, e)) % n;
        System.out.println("Encrypted message is : " + c);
        BigInteger N = BigInteger.valueOf(n);
        BigInteger C = BigDecimal.valueOf(c).toBigInteger();
        msgback = (C.pow(d)).mod(N);
        System.out.println("Decrypted message is : "+ msgback);
    }
    static int gcd(int e, int z)
    {
        if (e == 0)
            return z;
        else
            return gcd(z % e, e);
    }
}
```

```
C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java RSA.java
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12

C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>
```

6.LeakyBucket

```
import java.util.*;

import java.io.*;

public class LeakyBucket {

	public static void main (String[] args) {

		int no_of_queries,storage,output_pkt_size;

		int input_pkt_size,bucket_size,size_left;

		storage=0;

		no_of_queries=4;

		bucket_size=10;

		input_pkt_size=4;

		output_pkt_size=1;

		for(int i=0;i<no_of_queries;i++)

		{

			size_left=bucket_size-storage;


			if(input_pkt_size<=(size_left))

			{

				storage+=input_pkt_size;

				System.out.println("Buffer size= "+storage+

					" out of bucket size= "+bucket_size);
```

```
                }
                else
                {
                        System.out.println("Packet loss = "
                                        +(input_pkt_size-(size_left)));
                        storage=bucket_size;
                        System.out.println("Buffer size= "+storage+
                                        " out of bucket size= "+bucket_size);
                }
                storage-=output_pkt_size;
            }
        }
}
```

```
C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>java LeakyBucket.java
Buffer size= 4 out of bucket size= 10
Buffer size= 7 out of bucket size= 10
Buffer size= 10 out of bucket size= 10
Packet loss = 3
Buffer size= 10 out of bucket size= 10

C:\Users\WIN10\OneDrive\Desktop\CN-lab-programs-main>
```