**Part –B**

1. **Write a program for error detecting code using CRC-CCITT (16- bits).**

import java.io.\*;   
import java.util.Scanner;  
  
class crcscanner  
{  
    public static void main(String a[]) throws IOException  
    {  
          
          Scanner sc=new Scanner(System.in);  
          int[] message;  
          int[] gen;  
          int[] app\_message;  
          int[] rem;  
          int[] trans\_message;  
          int message\_bits,gen\_bits, total\_bits;  
          
         System.out.println("\n Enter number of bits in message : ");  
         message\_bits=sc.nextInt();  
   
         message=new int[message\_bits];  
         System.out.println("\n Enter message bits : ");  
         for(int i=0; i<message\_bits; i++)  
         message[i]=sc.nextInt();  
        System.out.println("\n Enter number of bits in gen : ");  
        gen\_bits=sc.nextInt();  
          
        gen=new int[gen\_bits];  
        System.out.println("\n Enter gen bits : ");  
        for(int i=0; i<gen\_bits; i++)  
       {  
         gen[i]=sc.nextInt();  
       }  
  
  
      total\_bits=message\_bits+gen\_bits-1;   
     
      app\_message=new int[total\_bits];  
      rem=new int[total\_bits];  
      trans\_message=new int[total\_bits];  
          
      for(int i=0;i<message.length;i++)  
      {  
     app\_message[i]=message[i];  
      }  
             
      System.out.print("\n Message bits are : ");  
      for(int i=0; i< message\_bits; i++)  
      {  
  System.out.print(message[i]);          
      }   
      System.out.print("\n Generators bits are : ");  
      for(int i=0; i< gen\_bits; i++)  
      {  
    System.out.print(gen[i]);          
      }           
      System.out.print("\n Appended message is : ");  
      for(int i=0; i< app\_message.length; i++)  
      {  
 System.out.print(app\_message[i]);          
      }  
          
          
      for(int j=0; j<app\_message.length; j++)  
     {  
              rem[j] = app\_message[j];  
     }  
      
     rem=computecrc(app\_message, gen, rem);  
          
     for(int i=0;i<app\_message.length;i++)          
     {  
           trans\_message[i]=(app\_message[i]^rem[i]);  
     }  
          
     System.out.println("\n Transmitted message from the transmitter is : ");      
     for(int i=0;i<trans\_message.length;i++)  
     {   
    System.out.print(trans\_message[i]);  
     }  
              
     System.out.println("\n Enter received message of "+total\_bits+" bits at receiver end : ");  
     for(int i=0; i<trans\_message.length; i++)  
     {  
      trans\_message[i]=sc.nextInt();;  
     }  
     System.out.println("\n Received message is :");    
     for(int i=0; i< trans\_message.length; i++)  
     {  
     System.out.print(trans\_message[i]);          
     }  
             
     for(int j=0; j<trans\_message.length; j++)  
    {  
              rem[j] = trans\_message[j];  
    }  
    rem=computecrc(trans\_message, gen, rem);  
     for(int i=0; i< rem.length; i++)  
    {  
          if(rem[i]!=0)   
   
                
          {  
                System.out.println("\n There is Error in the received me            ");  
                break;  
          }  
          if(i==rem.length-1)  
            
     System.out.println("\n There is No Error in the received m ");  
     }  
  }  
  
   static int[] computecrc(int app\_message[],int gen[], int rem[])  
 {  
      int current=0;  
      while(true)  
     {  
            for(int i=0;i<gen.length;i++)  
           {  
      rem[current+i]=(rem[current+i]^gen[i]);  
           }  
            while(rem[current]==0 && current!=rem.length-1)  
          {  
      current++;  
 }  
            if((rem.length-current)<gen.length)  
      {  
           break;  
 }  
     }  
     return rem;  
 }  
}

1. **Write a program to find the shortest path between vertices using bellman-ford algorithm.**

import java.util.Scanner;

public class BellmanFord

{

private int D[]; private int num\_ver;

public static final int MAX\_VALUE = 999;

public BellmanFord(int num\_ver)

{

this.num\_ver = num\_ver; D = new int[num\_ver + 1];

}

public void BellmanFordEvaluation(int source, int A[][])

{

for (int node = 1; node <= num\_ver; node++)

{

D[node] = MAX\_VALUE;

}

D[source] = 0;

for (int node = 1; node <= num\_ver - 1; node++)

{

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

if (A[sn][dn] != MAX\_VALUE)

{

if (D[dn] > D[sn]+ A[sn][dn])

D[dn] = D[sn] + A[sn][dn];

}

}

}

}

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

if (A[sn][dn] != MAX\_VALUE)

{

if (D[dn] > D[sn]+ A[sn][dn])

System.out.println("The Graph contains negative egde cycle");

}

}

}

for (int vertex = 1; vertex <= num\_ver; vertex++)

{

System.out.println("distance of source " + source + " to "+ vertex + " is " + D[vertex]);

}

}

public static void main(String[ ] args)

{

int num\_ver = 0; int source;

Scanner scanner = new Scanner(System.in); System.out.println("Enter the number of vertices"); num\_ver = scanner.nextInt();

int A[][] = new int[num\_ver + 1][num\_ver + 1]; System.out.println("Enter the adjacency matrix"); for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

A[sn][dn] = scanner.nextInt(); if (sn == dn)

{

A[sn][dn] = 0; continue;

}

if (A[sn][dn] == 0)

{

A[sn][dn] = MAX\_VALUE;

}

}

}

System.out.println("Enter the source vertex"); source = scanner.nextInt();

BellmanFord b = new BellmanFord (num\_ver); b.BellmanFordEvaluation(source, A); scanner.close();

}

}

**Input graph:**

****

Output:



**9.Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.**

**Server Program**

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class SimpleFileServer {

public final static int SOCKET\_PORT = 13267; // you may change this

public final static String FILE\_TO\_SEND = "e:/source1.txt"; // you may change this

public static void main (String [] args ) throws IOException {

FileInputStream fis = null;

BufferedInputStream bis = null;

OutputStream os = null;

ServerSocket servsock = null;

Socket sock = null;

try {

servsock = new ServerSocket(SOCKET\_PORT);

while (true) {

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

try {

sock = servsock.accept();

System.out.println("Accepted connection : " + sock);

// send file

File myFile = new File (FILE\_TO\_SEND);

byte [] mybytearray = new byte [(int)myFile.length()];

fis = new FileInputStream(myFile);

bis = new BufferedInputStream(fis);

bis.read(mybytearray,0,mybytearray.length);

os = sock.getOutputStream();

System.out.println("Sending " + FILE\_TO\_SEND + "(" + mybytearray.length + " bytes)");

os.write(mybytearray,0,mybytearray.length);

os.flush();

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

}

finally {

if (bis != null) bis.close();

if (os != null) os.close();

if (sock!=null) sock.close();

}

}

}

finally {

if (servsock != null) servsock.close();

}

}

}

**Client Program**

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.Socket;

public class SimpleFileClient {

public final static int SOCKET\_PORT = 13267; // you may change this

public final static String SERVER = "127.0.0.1"; // localhost

public final static String

FILE\_TO\_RECEIVED = "e:/source-downloaded.txt"; // you may change this, I give a

// different name because i don't want to

// overwrite the one used by server...

public final static int FILE\_SIZE = 6022386; // file size temporary hard coded

// should bigger than the file to be downloaded

public static void main (String [] args ) throws IOException {

int bytesRead;

int current = 0;

FileOutputStream fos = null

BufferedOutputStream bos = null;

Socket sock = null;

try {

sock = new Socket(SERVER, SOCKET\_PORT);

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

// receive file

byte [] mybytearray = new byte [FILE\_SIZE];

InputStream is = sock.getInputStream();

fos = new FileOutputStream(FILE\_TO\_RECEIVED);

bos = new BufferedOutputStream(fos);

bytesRead = is.read(mybytearray,0,mybytearray.length);

current = bytesRead;

do {

bytesRead =

is.read(mybytearray, current, (mybytearray.length-current));

if(bytesRead >= 0) current += bytesRead;

} while(bytesRead > -1);

bos.write(mybytearray, 0 , current);

bos.flush();

System.out.println("File " + FILE\_TO\_RECEIVED

+ " downloaded (" + current + " bytes read)");

}

finally {

if (fos != null) fos.close();

if (bos != null) bos.close();

if (sock != null) sock.close();

}

}

}

**10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.**

**UDP Client**

import java.io.\*;

import java.net.\*;

public class UDPC

{

public static void main(String[] args)

{

DatagramSocket skt;

try {

skt=new DatagramSocket(); String msg= "text message "; byte[] b = msg.getBytes();

InetAddress host=InetAddress.getByName("127.0.0.1"); int serverSocket=6788;

DatagramPacket request =new DatagramPacket (b,b.length,host,serverSocket); skt.send(request);

byte[] buffer =new byte[1000];

DatagramPacket reply= new DatagramPacket(buffer,buffer.length); skt.receive(reply);

System.out.println("client received:" +new String(reply.getData())); skt.close();

}

catch(Exception ex)

{

}

}

}

**UDP Server**

import java.io.\*; import java.net.\*;

public class UDPS

{

public static void main(String[] args)

{

DatagramSocket skt=null;

try

{

skt=new DatagramSocket(6788); byte[] buffer = new byte[1000];

while(true)

{

DatagramPacket request = new DatagramPacket(buffer,buffer.length);

skt.receive(request);

String[] message = (new String(request.getData())).split("");

byte[] sendMsg= (message[1]+ " server processed").getBytes();

DatagramPacket reply = new DatagramPacket(sendMsg,sendMsg.length,request.getAddress(),request.getPort());

skt.send(reply);

}

}

catch(Exception ex)

{

}

}

}

1. **Write a program for simple RSA algorithm to encrypt and decrypt the data.**

Implementation of RSA Algorithm(Encryption and Decryption) in Java

import java.math.BigInteger;

import java.util.Random;

import java.io.\*;

public class RSA {

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private int blocksize = 256;

//blocksize in byte

private Random r;

public 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);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0 ) {

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

public RSA(BigInteger e, BigInteger d, BigInteger N) {

this.e = e;

this.d = d;

this.N = N;

}

public static void main (String[] args) throws IOException {

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()));

// encrypt

byte[] encrypted = rsa.encrypt(teststring.getBytes());

System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted));

// decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypted String in Bytes: " + bytesToString(decrypted));

System.out.println("Decrypted String: " + new String(decrypted));

}

private static String bytesToString(byte[] encrypted) {

String test = "";

for (byte b : encrypted) {

test += Byte.toString(b);

}

return test;

}

//Encrypt message

public byte[] encrypt(byte[] message) {

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

// Decrypt message

public byte[] decrypt(byte[] message) {

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}

1. **Write a program for congestion control using leaky bucket algorithm.**

filename:Licky.java

import java.io.\*;

import java.util.\*;

class Queue

{

int q[],f=0,r=0,size;

void insert(int n)

{

Scanner in = new Scanner(System.in);

q=new int[10];

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

{

System.out.print("\nEnter " + i + " element: ");

int ele=in.nextInt();

if(r+1>10)

{

System.out.println("\nQueue is full \nLost Packet: "+ele);

break;

}

else

{

r++;

q[i]=ele;

}

}

}

void delete()

{

Scanner in = new Scanner(System.in);

Thread t=new Thread();

if(r==0)

System.out.print("\nQueue empty ");

else

{

for(int i=f;i<r;i++)

{

try

{

t.sleep(1000);

}

catch(Exception e){}

System.out.print("\nLeaked Packet: "+q[i]);

f++;

}

}

System.out.println();

}

}

class Licky extends Thread

{

public static void main(String ar[])throws Exception

{

Queue q=new Queue();

Scanner src=new Scanner(System.in);

System.out.println("\nEnter the packets to be sent:");

int size=src.nextInt();

q.insert(size);

q.delete();

}

}

/\*

OUTPUT

bash-3.00$ javac Licky.java

bash-3.00$ java Licky

Enter the packets to be sent:

11

Enter 0 element: 1

Enter 1 element: 0

Enter 2 element: 2

Enter 3 element: 3

Enter 4 element: 4

Enter 5 element: 5

Enter 6 element: 6

Enter 7 element: 7

Enter 8 element: 8

Enter 9 element: 9

Enter 10 element: 10

Queue is full

Lost Packet: 10

Leaked Packet: 1

Leaked Packet: 0

Leaked Packet: 2

Leaked Packet: 3

Leaked Packet: 4

Leaked Packet: 5

Leaked Packet: 6

Leaked Packet: 7

Leaked Packet: 8

Leaked Pa