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

import java.util.Scanner;

public class crc {

public String crc (String dividend, String divisor)

{

String str1, div;

int shift;

shift = dividend.length( ) - divisor.length( );

while (shift >= 0)

{

//XORing the string

dividend = Integer.*toBinaryString* (Integer.*parseInt*(dividend, 2 ) ^

(Integer.*parseInt* (divisor, 2) << shift));

shift = dividend.length( ) - divisor.length( );

}

if (dividend.length( ) < 16)

{

while (dividend.length( ) != 16)

{

dividend = "0" + dividend;

}

}

System.*out*.println("Div="+dividend);

return dividend;

}

public static void main (String args[ ])

{

String data, checksum, syn, dividend, Received\_data;

int padding;

String polynomial = "10001000000100001";

Scanner sc = new Scanner(System.*in*);

//Data to be transmitted from sender side

System.*out*.println("Enter the data to be encrypted\n");

data = sc.next( );

dividend = data;

padding = polynomial.length( ) - 1;

//Zero padding of data based on polynomial

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

{

dividend += "0";

}

crc obj = new crc( );

checksum = obj.crc(dividend, polynomial);

//Generated Codeword

data = data + checksum;

System.*out*.println("Sender Checksum="+checksum);

System.*out*.println( "Code word transmitted overnetwork="+data);

//Data received at the receiver side

System.*out*.println("Enter the received codeword\n");

Received\_data = sc.next( );

syn = obj.crc(Received\_data,polynomial);

//Generated Syn bits after checking checksum

if(Long.*parseLong* (syn) == 0)

System.*out*.println("No error in data transmission");

else

System.*out*.println("Error in transmission");

} }

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

import java.io.\*;

import java.util.Scanner;

class dist

{

public static void main(String args[])

{

int dmat[][];

int dist[][];

int via[][];

int n=0,i=0,j=0,k=0,count=0;

Scanner in = new Scanner(System.in);

System.out.println("enter the number of nodes\n");

n = in.nextInt();

dmat = new int[n][n];

dist = new int[n][n];

via = new int[n][n];

System.out.println("enter the cost matrix\n");

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

for(j=0;j<n;j++)

{

dmat[i][j] = in.nextInt();

dmat[i][i]=0;

dist[i][j]=dmat[i][j];

via[i][j]=j;

}

do

{

count=0;

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

for(j=0;j<n;j++)

for(k=0;k<n;k++)

if(dist[i][j]>dmat[i][k]+dist[k][j])

{

dist[i][j]=dist[i][k]+dist[k][j];

via[i][j]=k;

count++;

}

}

while(count!=0);

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

{

System.out.println("state value for router"+i+" is");

for(j=0;j<n;j++)

{

System.out.println("To "+j+" -Via "+via[i][j]+" distance is "+dist[i][j]);

}

}

}

}

3 .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 Side:**

import java.net.\*;

import java.io.\*;

public class ContentsServer

{

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

{ // establishing the connection with the server

ServerSocket sersock = new ServerSocket(6002);

System.out.println("Server ready for connection");

Socket sock = sersock.accept(); // binding with port: 6002

System.out.println("Connection is successful and wating for chatting");

// reading the file name from client

InputStream istream = sock.getInputStream( );

BufferedReader fileRead =new BufferedReader(new

InputStreamReader(istream));

String fname = fileRead.readLine( );

// reading file contents

BufferedReader contentRead = new BufferedReader(new

FileReader(fname) );

// keeping output stream ready to send the contents

OutputStream ostream = sock.getOutputStream( );

PrintWriter pwrite = new PrintWriter(ostream, true);

String str;

while((str = contentRead.readLine()) != null) // reading line-by-line from file

{

pwrite.println(str); // sending each line to client

}

sock.close( ); sersock.close( ); // closing network sockets

pwrite.close( ); fileRead.close( ); contentRead.close( );

}

}

**Client Side:**

import java.net.\*;

import java.io.\*;

public class ContentsClient

{

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

{

@SuppressWarnings("resource")

Socket sock = new Socket( "127.0.0.1", 6002);

// reading the file name from keyboard. Uses input stream

System.out.print("Enter the file name");

BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

String fname = keyRead.readLine();

// sending the file name to server. Uses PrintWriter

OutputStream ostream = sock.getOutputStream( );

PrintWriter pwrite = new PrintWriter(ostream, true);

pwrite.println(fname);

// receiving the contents from server. Uses input stream

InputStream istream = sock.getInputStream();

BufferedReader socketRead = new BufferedReader(new

InputStreamReader(istream));

String str;

while((str = socketRead.readLine()) != null) // reading line-by-line

{

System.out.println(str);

}

pwrite.close( ); socketRead.close( ); keyRead.close( );

}

}

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

**Client Side:**

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

**class** UDPClient

{

**public** **static** **void** main(String args[ ]) **throws** Exception

{

DatagramSocket clientSocket=**new** DatagramSocket( );

InetAddress IPAddress=InetAddress.*getByName*("127.0.0.1");

**byte**[] sendData=**new** **byte**[10];

**byte**[] receiveData=**new** **byte**[10];

String sentence= "Hi, I am Client. Send me a message";

sendData=sentence.getBytes( );

DatagramPacket sendPacket=**new** DatagramPacket(sendData,sendData.length, IPAddress,

11118);

clientSocket.send(sendPacket);

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

clientSocket.receive(receivePacket);

String reply=**new** String(receivePacket.getData( ));

System.*out*.println("From Server:" + reply);

clientSocket.close( );

}

}

**Server Side:**

import java.io.\*;

import java.net.\*;

class UDPServer

{

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

{

BufferedReader inFromUser=new BufferedReader(new InputStreamReader(System.in));

DatagramSocket serverSocket=new DatagramSocket(11118);

byte[ ] receiveData=new byte[10];

byte[ ] sendData=new byte[10];

while(true)

{

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

serverSocket.receive(receivePacket);

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

System.out.println("Client Message:"+sentence);

InetAddress IPAddress=receivePacket.getAddress( );

int port=receivePacket.getPort( );

String message=inFromUser.readLine( );

sendData=message.getBytes( );

DatagramPacket sendPacket=new DatagramPacket(sendData, sendData.length,

IPAddress, port);

serverSocket.send(sendPacket);

}

}

}

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

import java.util.\*;

import java.math.BigInteger;

import java.lang.\*;

class RSA

{

public static void main(String[ ] args)

{

Random rand1=new Random(System.currentTimeMillis( ));

Random rand2=new Random(System.currentTimeMillis( )\*10);

int pubkey=2;

BigInteger p=BigInteger.probablePrime(32, rand1);

BigInteger q=BigInteger.probablePrime(32, rand2);

BigInteger n=p.multiply(q);

BigInteger p\_1=p.subtract(new BigInteger("1"));

BigInteger q\_1=q.subtract(new BigInteger("1"));

BigInteger z=p\_1.multiply(q\_1);

while(true)

{

BigInteger GCD=z.gcd(new BigInteger(""+pubkey));

if(GCD.equals(BigInteger.ONE))

{

break;

}

pubkey++;

}

BigInteger big\_pubkey=new BigInteger(""+pubkey);

BigInteger prvkey=big\_pubkey.modInverse(z);

System.out.println("public key : "+big\_pubkey+","+n);

System.out.println("private key : "+prvkey+","+n);

//RSA Encryption and Decryption

Scanner sc = new Scanner(System.in);

System.out.println("Enter the message to be encrypted");

String msg = sc.nextLine( );

byte[] bytes = msg.getBytes( );

for(int i=0;i<msg.length( );i++)

{

int asciiVal=bytes[i];

BigInteger val=new BigInteger(""+asciiVal);

BigInteger cipherVal=val.modPow(big\_pubkey,n);

System.out.println("Cipher text: " +cipherVal);

BigInteger plainVal=cipherVal.modPow(prvkey,n);

int i\_plainVal=plainVal.intValue();

System.out.println("Plain text:"+Character.toString((char)i\_plainVal));

}

}

}

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

**import** java.util.\*;

**public** **class** leaky

{

**public** **static** **void** main(String[ ] args)

{

**int** time, output\_rate, max\_buffer\_size, num\_of\_pkts, count=0, cur\_buffer\_size=0;

Scanner in = **new** Scanner(System.*in*);

System.*out*.println("Enter the maximum size of buffer");

max\_buffer\_size = in.nextInt( );

System.*out*.println("Enter the output rate of packets from the buffer");

output\_rate = in.nextInt( );

System.*out*.println("Enter the number of arriving packets");

num\_of\_pkts = in.nextInt( );

**int**[] pkt\_size = **new** **int**[num\_of\_pkts];

**int**[] arr\_time\_of\_pkts =**new** **int**[num\_of\_pkts];

System.*out*.println("Enter the time of arrival of packets");

**for**(count=0;count<num\_of\_pkts;count++)

{

arr\_time\_of\_pkts[count] = in.nextInt( );

}

time=0;

count=0;

**while**(count <num\_of\_pkts)

{

**if**(time==arr\_time\_of\_pkts[count])

{

Random rn = **new** Random();

pkt\_size[count] = (rn.nextInt(10)+1) \* 10;

System.*out*.println("Packet "+(count+1)+" has arrived & its size is:" +pkt\_size[count]);

System.*out*.println("Current Size of buffer:"+cur\_buffer\_size);

**if** (cur\_buffer\_size + pkt\_size [count] <=max\_buffer\_size)

{

cur\_buffer\_size += pkt\_size[count];

System.*out*.println("Packet"+(count+1)+" arriving at "+arr\_time\_of\_pkts[count]+ "is CONFORMING PACKET\n");

}

**else**

{

System.*out*.println("Packet "+(count+1)+" arriving at "+arr\_time\_of\_pkts[count]+" is NON CONFORMING PACKET asit exceeds the buffer limit\n");

}

count++;

}

time++;

cur\_buffer\_size -= output\_rate;

**if**(cur\_buffer\_size< 0)

cur\_buffer\_size=0;

}

}

}