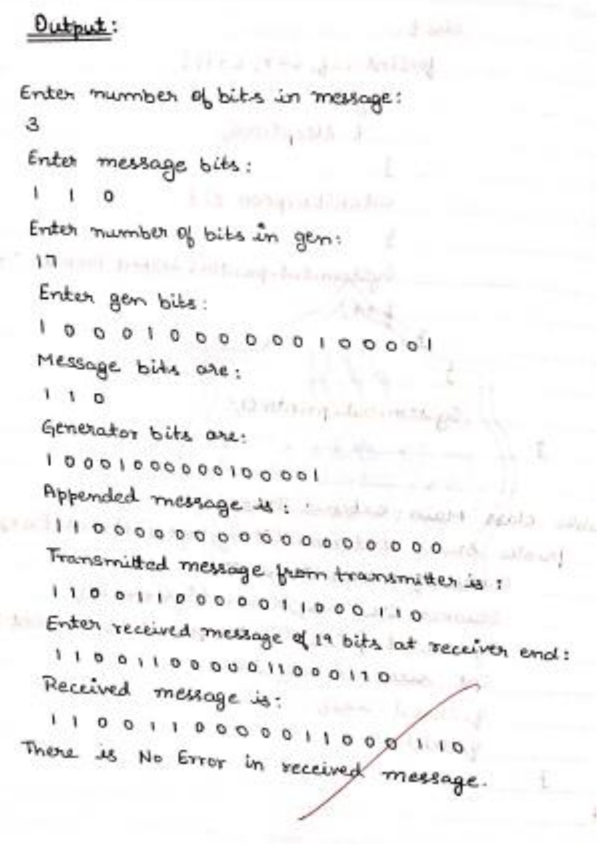
CN LAB

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

CRC

```
import java.io.*;
import java.util.Scanner;
class CRC
{
		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;
        }
}
```

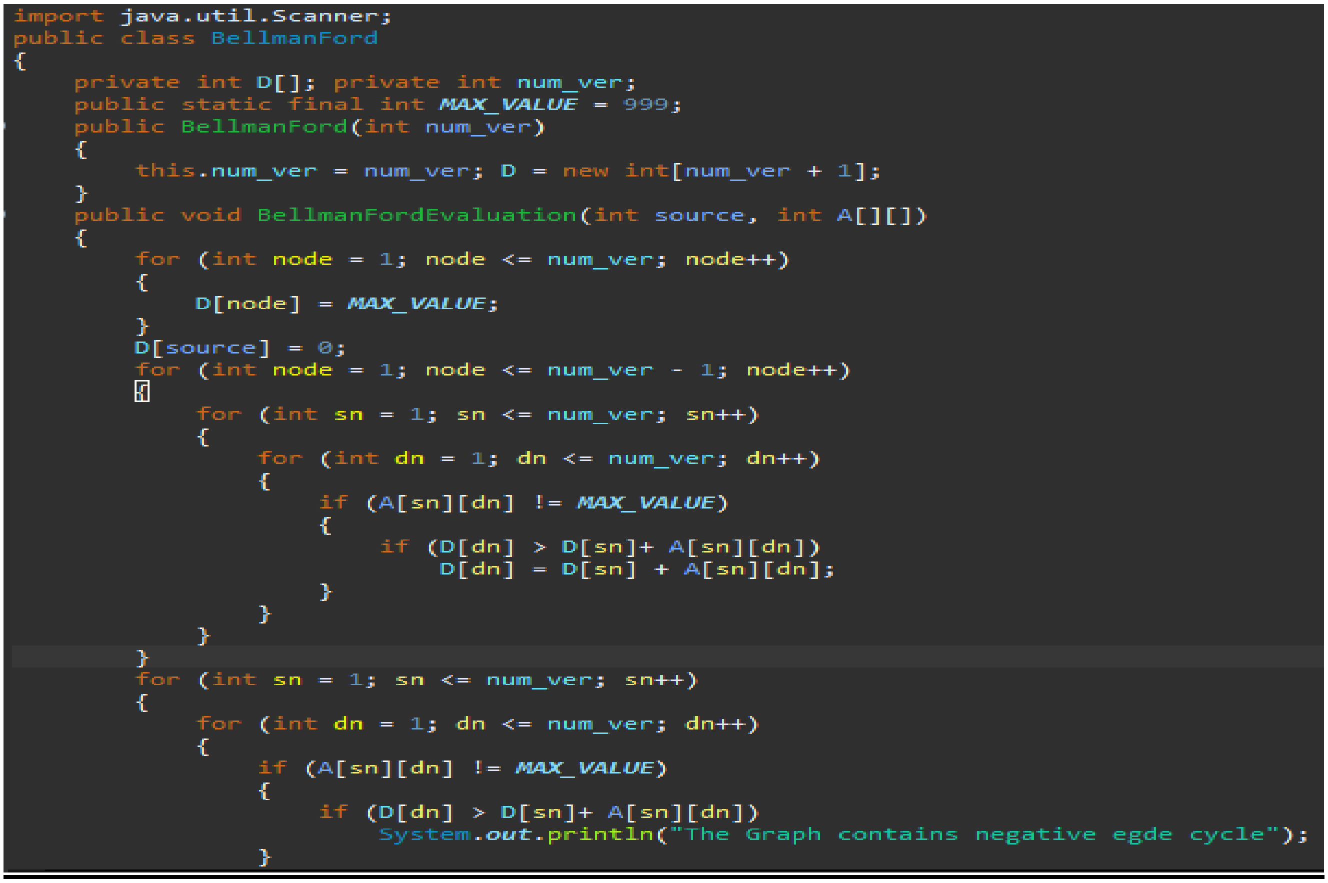


Output:

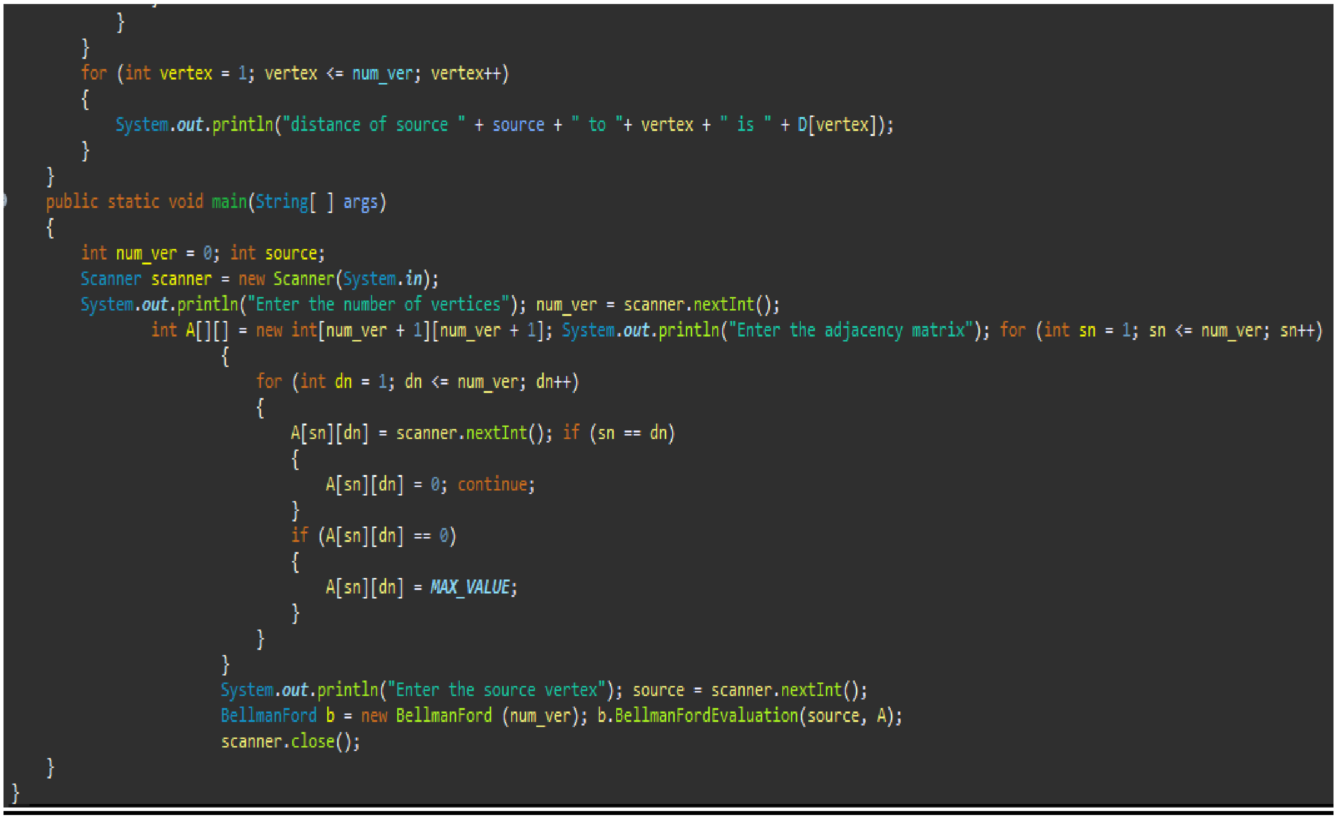
```
Enter number of bits in message:
3
Enter message bits:
1 1 0
Enter number of bits in gen:
17
Enter gen bits:
1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1
Message bits are:
1 1 0
Generator bits are:
10001000000100001
Appended message is :
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Transmitted message from transmitter is :
1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0
Enter received message of 19 bits at receiver end:
1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0
Received message is:
1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0
There is No Error in received message.
```

Program-2: **Write a program to find the shortest path between vertices using bellman-ford algorithm.**

BellmanFord



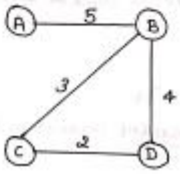
```
import java.util.Scanner;
public class BellmanFord
{
    private int D[]; private int num_ver;
    public static final int MAX_VALUE = 999;
    public BellmanFord(int num_ver)
    {
        this.num_ver = num_ver; D = new int[num_ver + 1];
    }
    public void BellmanFordEvaluation(int source, int 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();
    }
}
```

Output:

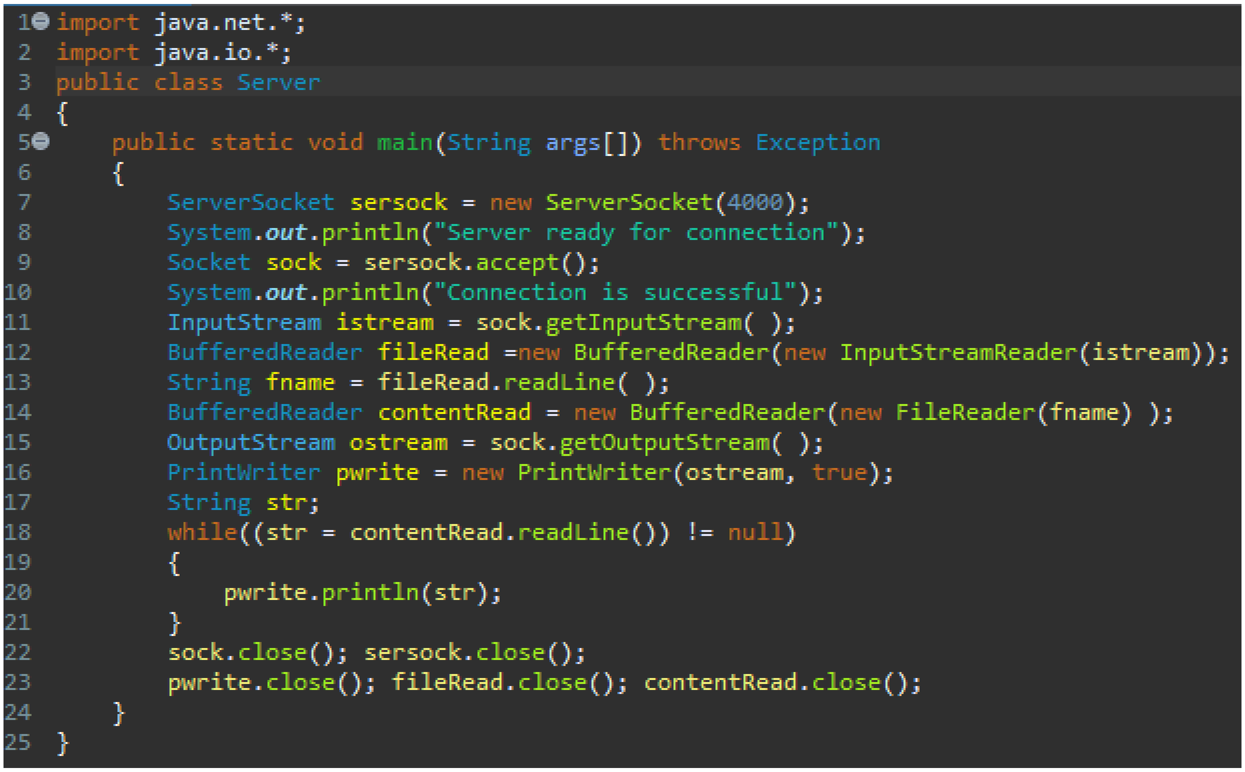
Input Graph:



```
Enter number of vertices
 4
Enter adjacency matrix
 0   5   0   0
 5   0   3   4
 0   3   0   2
 0   4   2   0

Enter source vertex
2
Distance of source 2 to 1 is 5
Distance of source 2 to 2 is 0
Distance of source 2 to 3 is 3
Distance of source 2 to 4 is 4
```

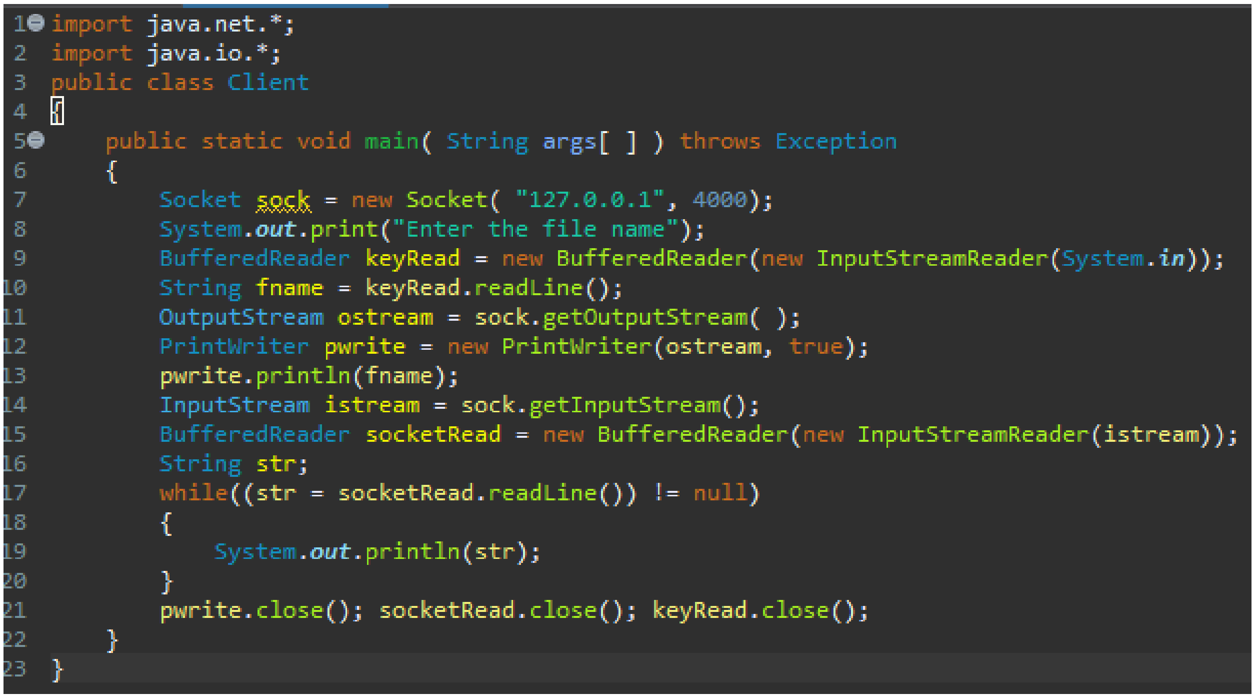
Program-3: Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Server

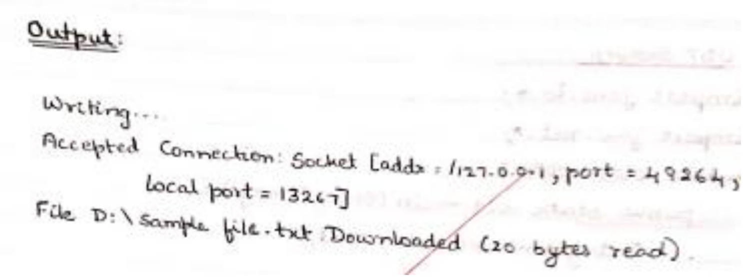


```java
import java.net.*;
import java.io.*;
public class Server
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket sersock = new ServerSocket(4000);
        System.out.println("Server ready for connection");
        Socket sock = sersock.accept();
        System.out.println("Connection is successful");
        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();
    }
}
```

Client



```java
import java.net.*;
import java.io.*;
public class Client
{
    public static void main( String args[ ] ) throws Exception
    {
        Socket sock = new Socket( "127.0.0.1", 4000);
        System.out.print("Enter the file name");
        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();
    }
}
```



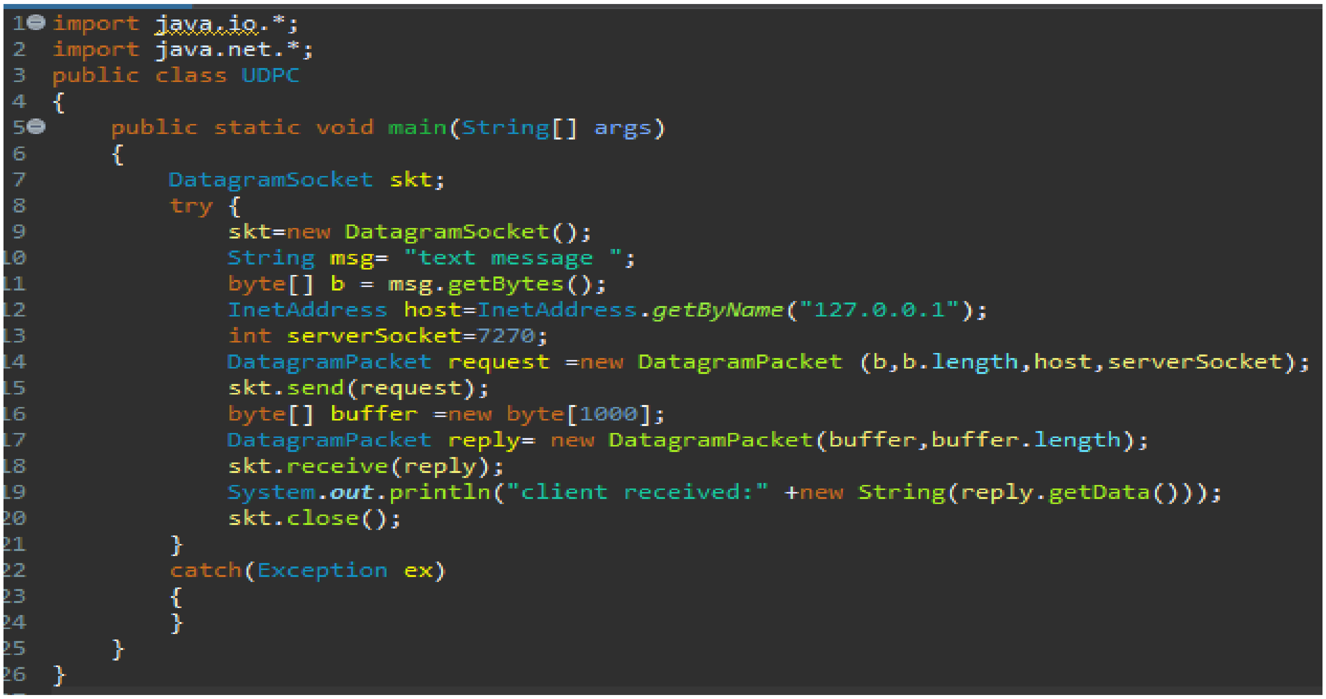
Output:

```
Writing...
Accepted Connection: Socket [addr = /127.0.0.1, port = 49264,
            local port = 13267]
File D:\ Sample file.txt Downloaded (20 bytes read).
```

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

Client

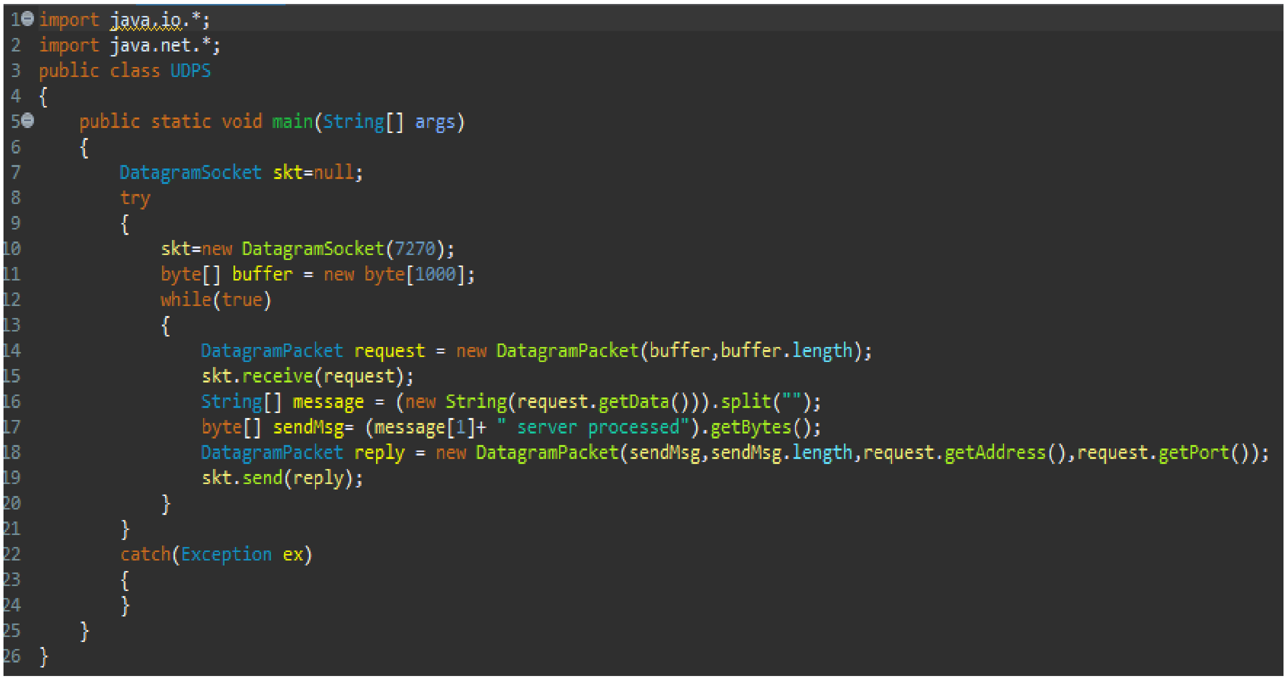
UDPC



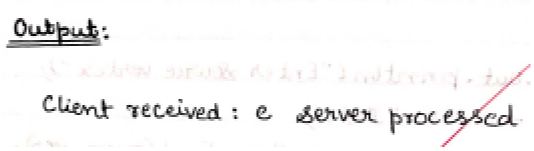
```java
import java.io.*;
import java.net.*;
public class UDPC
{
    public static void main(String[] args)
    {
        DatagramSocket skt;
        try {
            skt=new DatagramSocket();
            String msg= "text message ";
            byte[] b = msg.getBytes();
            InetAddress host=InetAddress.getByName("127.0.0.1");
            int serverSocket=7270;
            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)
        {
        }
    }
}
```

Server

UDPS



```
import java.io.*;
import java.net.*;
public class UDPS
{
    public static void main(String[] args)
    {
        DatagramSocket skt=null;
        try
        {
            skt=new DatagramSocket(7270);
            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)
        {
        }
    }
}
```



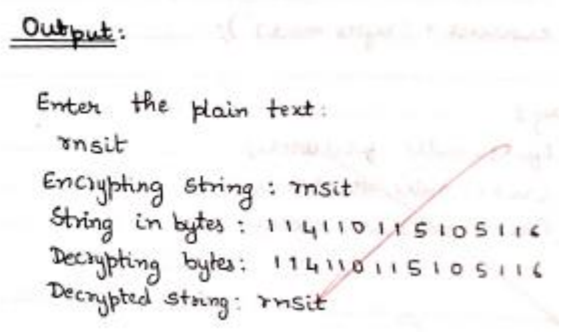
Output:

```
Client received : e server processed
```

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

RSA

```
import java.io.IOException;
import java.io.DataInputStream;
import java.math.BigInteger;
import java.util.Random;
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 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;
    }
    @SuppressWarnings("deprecation")
    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()));
        byte[] encrypted = rsa.encrypt(teststring.getBytes());
        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypting 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;
    }
    public byte[] encrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }
    public byte[] decrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```



Output:

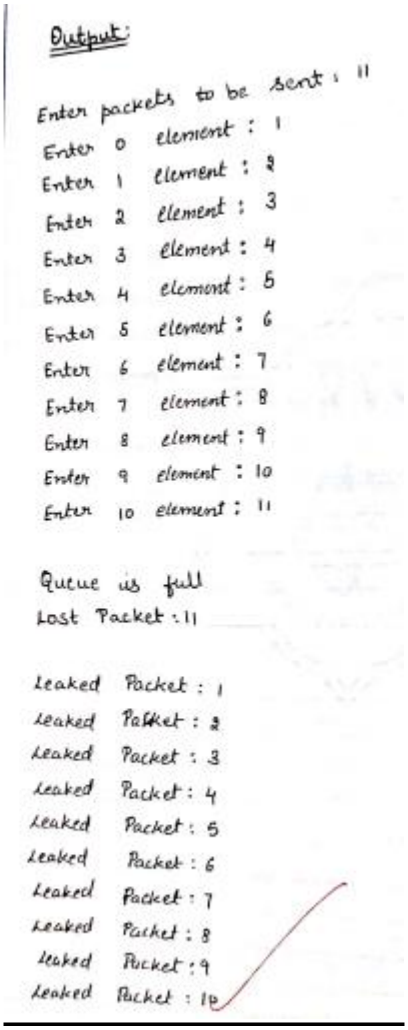
```
Enter the plain text:
 rnsit
Encrypting String : rnsit
String in bytes : 114110115105116
Decrypting bytes: 114110115105116
Decrypted string: rnsit
```

Program-6: Write a program for congestion control using leaky bucket algorithm.

Main

```
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();
}
}
public class Main extends Thread
```

```
{
    public static void main(String arg[])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:

```
Enter packets to be sent : 11
Enter 0 element : 1
Enter 1 element : 2
Enter 2 element : 3
Enter 3 element : 4
Enter 4 element : 5
Enter 5 element : 6
Enter 6 element : 7
Enter 7 element : 8
Enter 8 element : 9
Enter 9 element : 10
Enter 10 element : 11

Queue is full
Lost Packet : 11

Leaked Packet : 1
Leaked Packet : 2
Leaked Packet : 3
Leaked Packet : 4
Leaked Packet : 5
Leaked Packet : 6
Leaked Packet : 7
Leaked Packet : 8
Leaked Packet : 9
Leaked Packet : 10
```

Program-1: Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

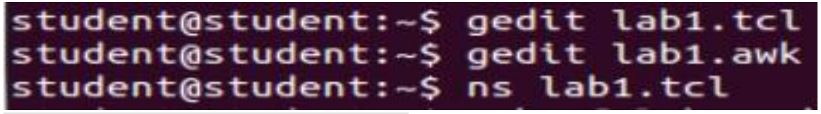
Step1: Open text editor, type the below program and save with extention .tcl (prog1.tcl)

```
set ns [new Simulator]
set nf [open prog1.nam w]
$ns namtrace-all $nf
set nd [open prog1.tr w]
$ns trace-all $nd
proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog1.nam &amp;
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512kb 10ms DropTail
$ns queue-limit $n1 $n2 10
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $udp0 $sink

$ns at 0.2 &quot;$cbr0 start&quot;
$ns at 4.5 &quot;$cbr0 stop&quot;
$ns at 5.0 &quot;finish&quot;
$ns run
```

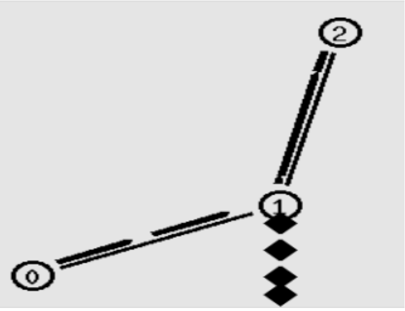
Step2: Open text editor, type the below program and save with extention .awk (prog1.awk)

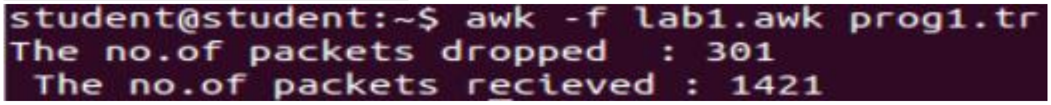
```
BEGIN {
dcount = 0;
rcount = 0;
}
{
event = $1;
if(event == &quot;d&quot;)
{
dcount++;
}
if(event == &quot;r&quot;)
{
rcount++;
}
}
END {
printf(&quot;The no.of packets dropped : %d\n &quot;,dcount);
printf(&quot;The no.of packets recieved : %d\n &quot;,rcount);
}
```

Output:



```
student@student:~$ gedit lab1.tcl
student@student:~$ gedit lab1.awk
student@student:~$ ns lab1.tcl
```





```
student@student:~$ awk -f lab1.awk prog1.tr
The no.of packets dropped  : 301
 The no.of packets recieved : 1421
```

Program-2: **Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.**

Step1: Open text editor, type the below program and save with extention .tcl (prog3.tcl)

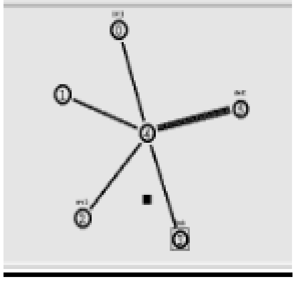
```
set ns [new Simulator]
set nf [open prog3.nam w]
$ns namtrace-all $nf
set nd [open prog3.tr w]
$ns trace-all $nd
proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog4.nam &amp;
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
$ns duplex-link $n1 $n0 1Mb 10ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 10ms DropTail
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts &quot;node [$node_ id] recieved ping answer from \
$from with round-trip-time $rtt ms.&quot;
}
set p1 [new Agent/Ping]
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]
$ns attach-agent $n1 $p1
$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6
$ns queue-limit $n0 $n4 3
$ns queue-limit $n0 $n5 2
$ns queue-limit $n0 $n6 2
$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6
$ns at 0.2 &quot;$p1 send&quot;
$ns at 0.4 &quot;$p2 send&quot;
$ns at 0.6 &quot;$p3 send&quot;
$ns at 1.0 &quot;$p4 send&quot;
```

```
$ns at 1.2 &quot;$p5 send&quot;
$ns at 1.4 &quot;$p6 send&quot;
$ns at 2.0 &quot;finish&quot;
$ns run
```

Step2: Open text editor, type the below program and save with extention .awk (prog3.awk)

```
BEGIN {
count=0;
}
{
event=$1;
if(event==&quot;d&quot;)
{
count++;
}
}
END {
printf(&quot;No of packets dropped : %d\n&quot;,count);
}
```

Output:



Program-3: **Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.**

Step1: Open text editor, type the below program and save with extention .tcl (prog5.tcl)

```
set ns [new Simulator]
set nf [open prog5.nam w]
$ns namtrace-all $nf
set nd [open prog5.tr w]
$ns trace-all $nd
$ns color 1 Blue
$ns color 2 Red
```

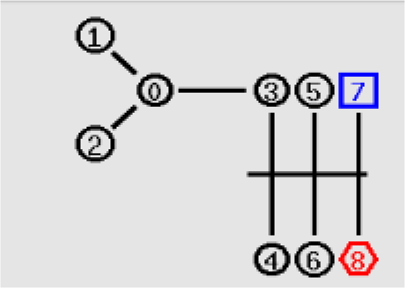
```
proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog5.nam &amp;
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red
$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail
$ns make-lan &quot;$n3 $n4 $n5 $n6 $n7 $n8&quot; 512Kb 40ms LL Queue/DropTail Mac/802_3
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right
$ns queue-limit $n0 $n3 20
set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1
$tcp1 set packetsize_ 55
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set tfile [open cwnd.tr w]
$tcp1 attach $tfile
$tcp1 trace cwnd_
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
```

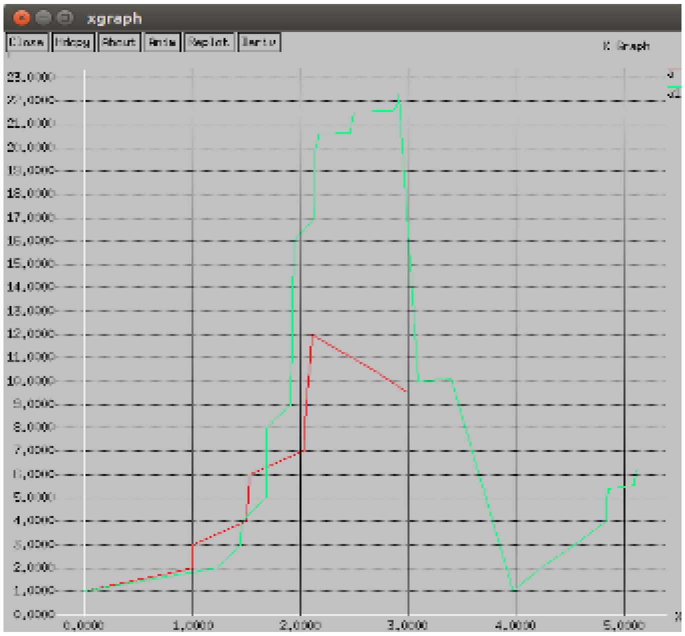
```
$ns connect $tcp2 $sink2
$tcp2 set class_ 2
$tcp2 set packetSize_ 55
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set tfile2 [open cwnd2.tr w]
$tcp2 attach $tfile2
$tcp2 trace cwnd_
$ns at 0.5 &quot;$ftp1 start&quot;
$ns at 1.0 &quot;$ftp2 start&quot;
$ns at 5.0 &quot;$ftp2 stop&quot;
$ns at 5.0 &quot;$ftp1 stop&quot;
$ns at 5.5 &quot;finish&quot;
$ns run
```

Step2: Open text editor, type the below program and save with extention .awk (prog5.awk)

```
BEGIN {
}
{
if($6==&quot;cwnd_&quot;) {
printf(&quot;%f\t%f\n&quot;,$1,$7);
}
}
END {
}
```

Output:





Program-4: **Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.**

Step1: Open text editor, type the below program and save with extention .tcl (prog6.tcl)

```
set ns [new Simulator]
set tf [open prog6.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open prog6.nam w]
$ns namtrace-all-wireless $nf 1000 1000
set val(chan) Channel/WirelessChannel ;
set val(prop) Propagation/TwoRayGround ;
set val(netif) Phy/WirelessPhy ;
set val(mac) Mac/802_11 ;
set val(ifq) Queue/DropTail/PriQueue ;
set val(ll) LL ;
set val(ant) Antenna/OmniAntenna ;
set val(ifqlen) 50 ;
set val(nn) 2 ;
set val(rp) AODV ;
set val(x) 500 ;
set val(y) 400 ;
set val(stop) 10.0 ;
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
create-god 3
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label &quot;tcp0&quot;
$n1 label &quot;sink1/tcp1&quot;
$n2 label &quot;sink2&quot;
$n0 set X_ 50
$n0 set Y_ 50
$n0 set Z_ 0
$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0
$n2 set X_ 600
$n2 set Y_ 600
$n2 set Z_ 0
$ns at 0.1 &quot;$n0 setdest 50 50 15&quot;
$ns at 0.1 &quot;$n1 setdest 100 100 25&quot;
$ns at 0.1 &quot;$n2 setdest 600 600 25&quot;
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2
```

```
$ns connect $tcp1 $sink2
$ns at 5 &quot;$ftp0 start&quot;
$ns at 5 &quot;$ftp1 start&quot;
$ns at 100 &quot;$n1 setdest 550 550 15&quot;
$ns at 190 &quot;$n1 setdest 70 70 15&quot;
proc finish { } {
global ns nf tf
$ns flush-trace
exec nam prog6.nam &amp;
close $tf
exit 0
}
$ns at 250 &quot;finish&quot;
$ns run
```

Step2: Open text editor, type the below program and save with extention .awk (prog6.awk)

```
BEGIN{
count1=0
count2=0
pack1=0
pack2=0
time1=0
time2=0
}
{
if($1=="r"&&$3=="_1_"&&$4=="AGT")
{
count1++
pack1=pack1+$8
time1=$2
}
f($1=="r"&&$3=="_2_"&&$4=="AGT")
{
 count2++
pack2=pack2+$8
time2=$2
 }
}
END{
printf("The Throughput from n0 to n1: %f Mbps \n", ((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2: %f Mbps", ((count2*pack2*8)/(time2*1000000)));
}
```