SHREE DEVI INSTITUTE OF TECHNOLOGY
(Affiliated to Visvesvaraya Technological University & Recognized by AICTE)
AIRPORT ROAD, KENJAR, MANGALORE – 574 142
Department of Computer Science and Engineering

B.E Computer Science & Engineering

LABORATORY MANUAL

COMPUTER NETWORK LABORATORY

(BCS502)

LAB IN-CHARGE:

Ms.Ananya J, Ms.Sowjanya P R

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

```
set ns [new Simulator]

set nf [open lab1.nam w]

$ns namtrace-all $nf

set tf [open lab1.tr w]

$ns trace-all $tf

proc finish { } {

global ns nf tf

$ns flush-trace

close $nf

close $tf

exec nam lab1.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Define labels for data flows

$n0 label "Source/udp0"

$n1 label "Source/udp1"

$n2 label "Router"
```

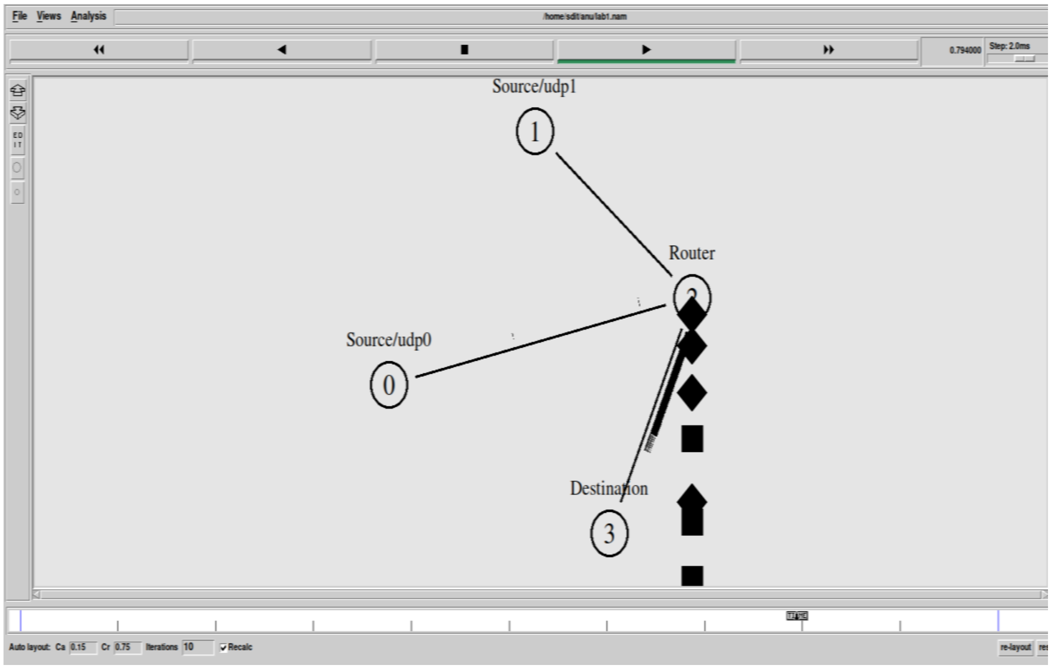
```
$n3 label "Destination"

$ns duplex-link $n0 $n2 200Mb 10ms DropTail

$ns duplex-link $n1 $n2 100Mb 5ms DropTail

$ns duplex-link $n2 $n3 1Mb 1000ms DropTail

$ns queue-limit $n0 $n2 10

$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 udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp1

set udp2 [new Agent/UDP]

$ns attach-agent $n2 $udp2

set cbr2 [new Application/Traffic/CBR]

$cbr2 attach-agent $udp2

set null0 [new Agent/Null]

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0
```

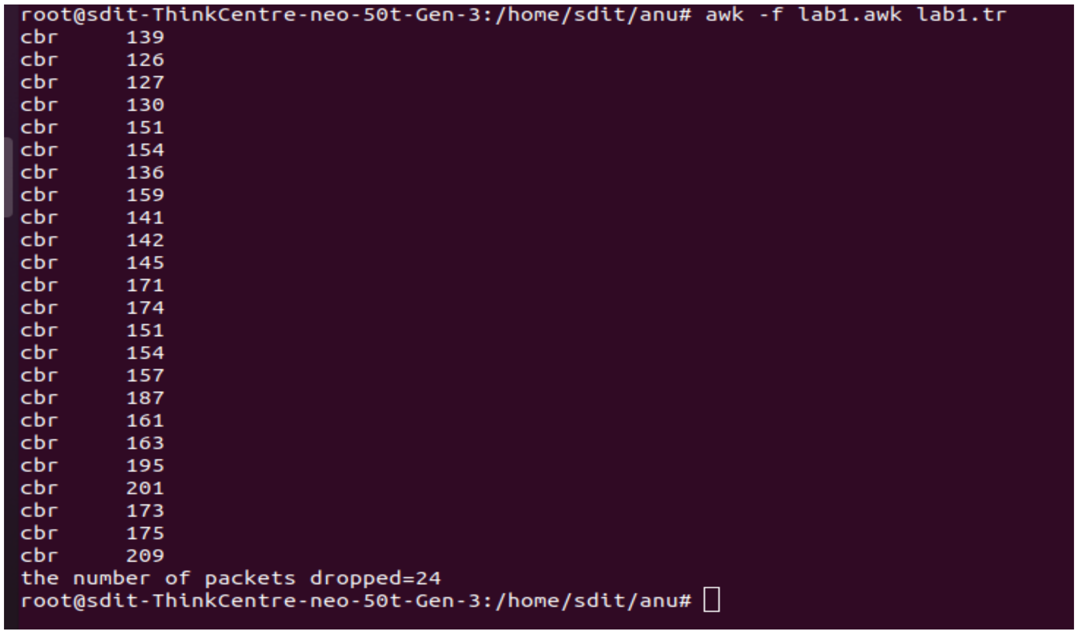
```
$ns connect $udp1 $null0

$ns at 0.1 "$cbr0 start"

$ns at 0.2 "$cbr1 start"

$ns at 1.0 "finish"

$ns run
```

AWK FILE:

```
BEGIN{ c=0;}

{

if($1== "d")

{

c++;

printf("%s\t%s\n",$5,$11);

}

}

END { printf("the number of packets dropped=%d\n",c);}
```

OUTPUT:





```
root@sdit-ThinkCentre-neo-50t-Gen-3:/home/sdit/anu# awk -f lab1.awk lab1.tr
cbr      139
cbr      126
cbr      127
cbr      130
cbr      151
cbr      154
cbr      136
cbr      159
cbr      141
cbr      142
cbr      145
cbr      171
cbr      174
cbr      151
cbr      154
cbr      157
cbr      187
cbr      161
cbr      163
cbr      195
cbr      201
cbr      173
cbr      175
cbr      209
the number of packets dropped=24
root@sdit-ThinkCentre-neo-50t-Gen-3:/home/sdit/anu# 
```

2. Aim: 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.

```
set ns [ new Simulator ]
set nf [ open lab2.nam w ]
$ns namtrace-all $nf
set tf [ open lab2.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [ $ns node ]
set n2 [ $ns node ]
set n3 [ $ns node ]
set n4 [ $ns node ]
set n5 [ $ns node ]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set p1 [ new Agent/Ping ]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [ new Agent/Ping ]
$ns attach-agent $n1 $p2
set p3 [new Agent/Ping ]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
set p4 [ new Agent/Ping ]
$ns attach-agent $n3 $p4
```

```
set p5 [ new Agent/Ping ]
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv { from rtt } {
$self instvar node_
puts "node [ $node_ id ] received answer from $from with round
trip time $rtt msec "
}
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab2.nam &
exit 0
}
$ns at 0.1 "$p1 send "
$ns at 0.2 "$p1 send "
$ns at 0.3 "$p1 send "
$ns at 0.4 "$p1 send "
$ns at 0.5 "$p1 send "
$ns at 0.6 "$p1 send "
$ns at 0.7 "$p1 send "
$ns at 0.8 "$p1 send "
$ns at 0.9 "$p1 send "
$ns at 1.0 "$p1 send "
```

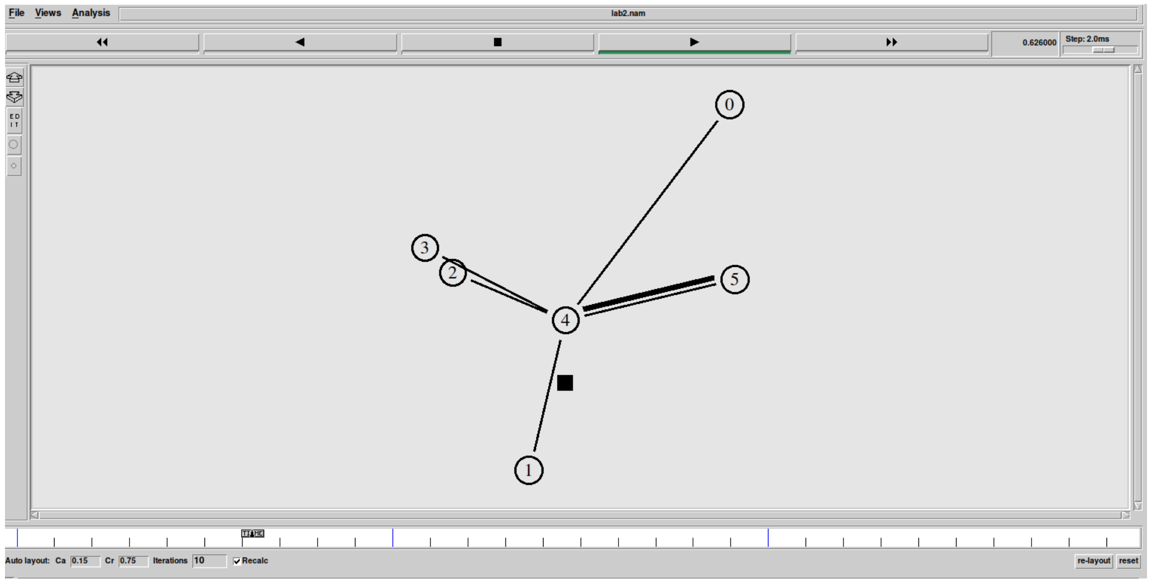
```
$ns at 1.1 "$p1 send "
$ns at 1.2 "$p1 send "
$ns at 1.3 "$p1 send "
$ns at 1.4 "$p1 send "
$ns at 1.5 "$p1 send "
$ns at 1.6 "$p1 send "
$ns at 1.7 "$p1 send "
$ns at 1.8 "$p1 send "
$ns at 1.9 "$p1 send "

$ns at 2.0 "$p1 send "
$ns at 2.1 "$p1 send "
$ns at 2.2 "$p1 send "
$ns at 2.3 "$p1 send "
$ns at 2.4 "$p1 send "
$ns at 2.5 "$p1 send "
$ns at 2.6 "$p1 send "
$ns at 2.7 "$p1 send "
$ns at 2.8 "$p1 send "
$ns at 2.9 "$p1 send "

$ns at 0.1 "$p3 send "
$ns at 1.1 "$p3 send "
$ns at 1.2 "$p3 send "
$ns at 1.3 "$p3 send "
$ns at 1.4 "$p3 send "
$ns at 1.5 "$p3 send "
$ns at 1.6 "$p3 send "
$ns at 1.7 "$p3 send "
$ns at 1.8 "$p3 send "
```

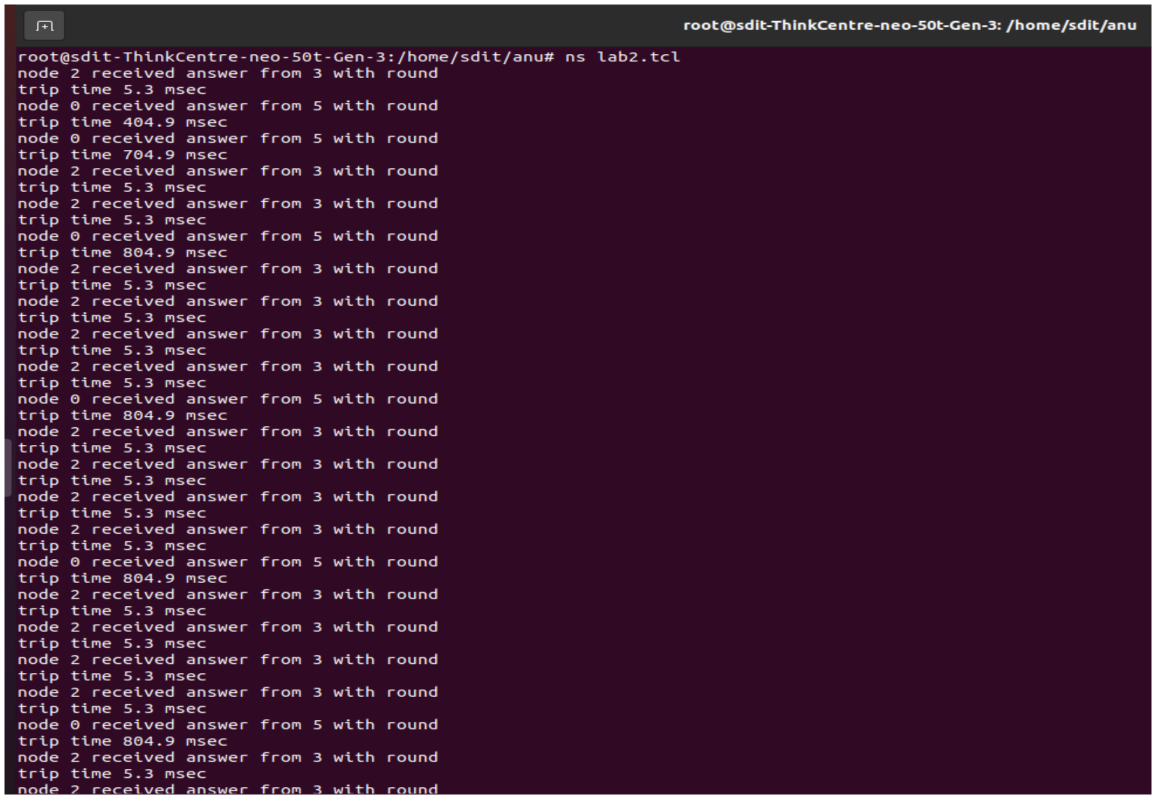
```
$ns at 1.9 "$p3 send "
$ns at 2.0 "$p3 send "

$ns at 2.1 "$p3 send "
$ns at 2.2 "$p3 send "
$ns at 2.3 "$p3 send "
$ns at 2.4 "$p3 send "
$ns at 2.5 "$p3 send "
$ns at 2.6 "$p3 send "
$ns at 2.7 "$p3 send "
$ns at 2.8 "$p3 send "
$ns at 2.9 "$p3 send "
$ns at 3.0 " finish "
$ns run
```

Awk file

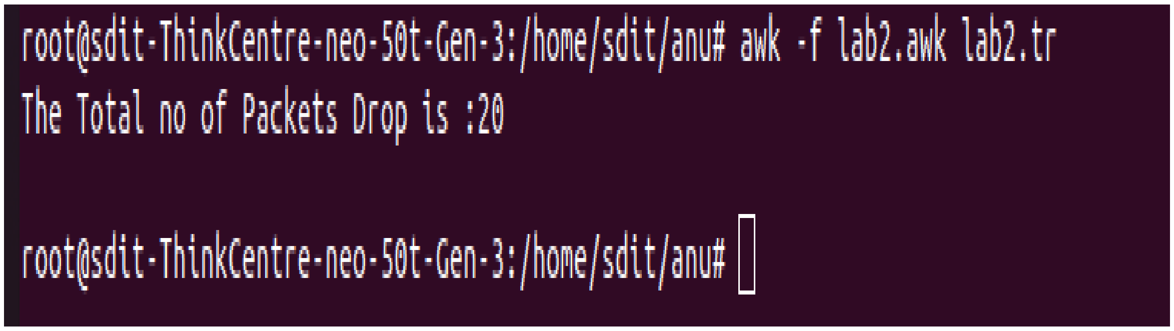
```
BEGIN{
count=0;
}
{
if($1=="d")
count++;
}
END{
printf("The Total no of Packets Drop is :%d\n\n", count);
}
```

OUTPUT:




root@sdit-ThinkCentre-neo-50t-Gen-3: /home/sdit/anu

```
root@sdit-ThinkCentre-neo-50t-Gen-3:/home/sdit/anu# ns lab2.tcl
node 2 received answer from 3 with round
trip time 5.3 msec
node 0 received answer from 5 with round
trip time 404.9 msec
node 0 received answer from 5 with round
trip time 704.9 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 0 received answer from 5 with round
trip time 804.9 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 0 received answer from 5 with round
trip time 804.9 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 0 received answer from 5 with round
trip time 804.9 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 0 received answer from 5 with round
trip time 804.9 msec
node 2 received answer from 3 with round
trip time 5.3 msec
node 2 received answer from 3 with round
```



```
root@sdit-ThinkCentre-neo-50t-Gen-3:/home/sdit/anu# awk -f lab2.awk lab2.tr
The Total no of Packets Drop is :20

root@sdit-ThinkCentre-neo-50t-Gen-3:/home/sdit/anu#
```

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

```
#Make a NS simulator
set ns [new Simulator]
set tf  [open lab3.tr w]
$ns trace-all $tf
set nf  [open lab3.nam w]
$ns namtrace-all $nf
# Create the nodes,color and label
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
$n1 color "red"
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
$n4 shape square
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
#Creates a lan from a set of nodes given by <nodelist>. Bandwidth, delay
#characteristics along with the link-layer, Interface queue, Mac layer and
#channel type for the lan also needs to be defined.
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 50Mb 100ms LL Queue/DropTail
# Create the link
```

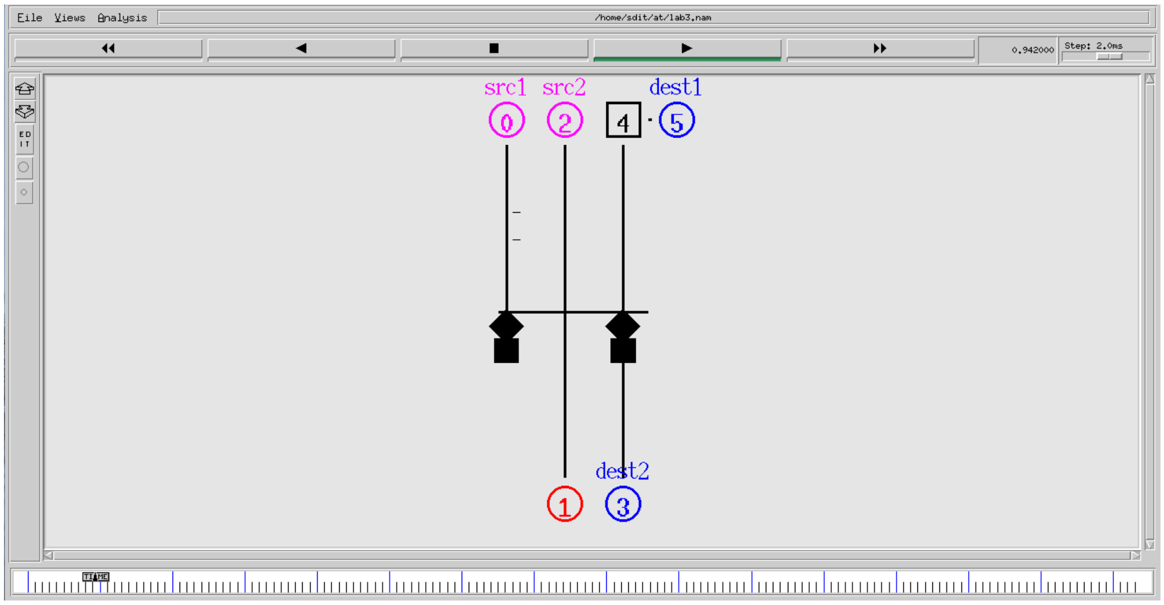
```
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
# Create the node position
$ns duplex-link-op $n4 $n5 orient right
# Add a TCP sending module to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
# Setup a FTP traffic generator on "tcp0"
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
# Add a TCP receiving module to node n5
set sink0 [new Agent/TCPSink]
$ns attach-agent $n5 $sink0
# Direct traffic from "tcp0" to "sink1"
$ns connect $tcp0 $sink0
# Add a TCP sending module to node n2
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 600
$ftp1 set interval_ 0.001
# Add a TCP receiving module to node n3
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1
set file1 [open file1.tr w]
$tcp0 attach $file1
```

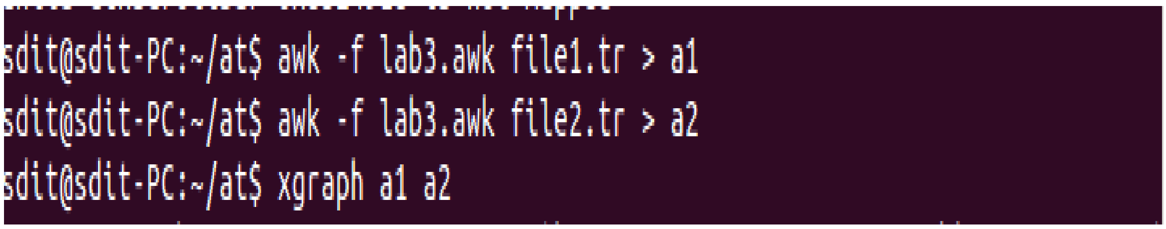
```
set file2 [open file2.tr w]
$tcp1 attach $file2
$tcp0 trace cwnd_
$tcp1 trace cwnd_
# Define a 'finish' procedure
proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam lab3.nam &
exit 0
}
# Schedule start/stop times
$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp1 start"
$ns at 15 "$ftp1 stop"
# Set simulation end time
$ns at 16 "finish"
$ns run
```

AWK FILE:

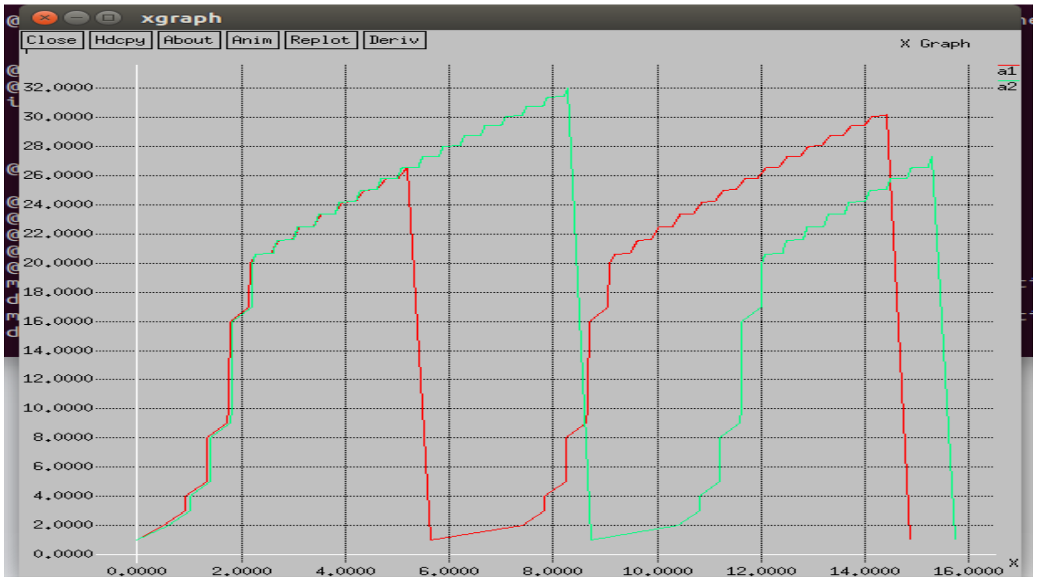
```
BEGIN{
}
{
if($6== "cwnd_")
printf( "%f\t%f\t\n",$1,$7);
}
END {
}
```

OUTPUT:





```
sdit@sdit-PC:~/at$ awk -f lab3.awk file1.tr > a1
sdit@sdit-PC:~/at$ awk -f lab3.awk file2.tr > a2
sdit@sdit-PC:~/at$ xgraph a1 a2
```


xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
X Graph
a1
a2
32.0000
30.0000
28.0000
26.0000
24.0000
22.0000
20.0000
18.0000
16.0000
14.0000
12.0000
10.0000
8.0000
6.0000
4.0000
2.0000
0.0000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000
12.0000
14.0000
16.0000
X

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

```
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.");

}

}
```

OUTPUT:

Enter message bits: 1101011011

Enter generator: 10011

The checksum code is: 11010110111110

Enter checksum code: 11010110111110

Enter generator: 10011

Data stream is valid

Enter message bits: 1101011011

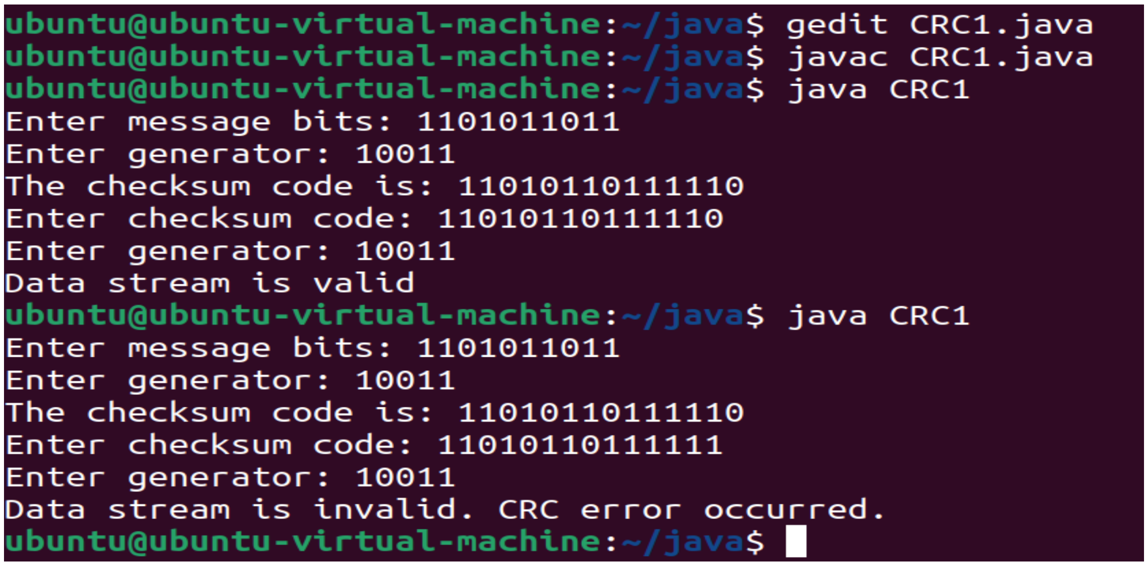
Enter generator: 10011

The checksum code is: 11010110111110

Enter checksum code: 11010110111111

Enter generator: 10011

Data stream is invalid. CRC error occurred.



```
ubuntu@ubuntu-virtual-machine:~/java$ gedit CRC1.java
ubuntu@ubuntu-virtual-machine:~/java$ javac CRC1.java
ubuntu@ubuntu-virtual-machine:~/java$ java CRC1
Enter message bits: 1101011011
Enter generator: 10011
The checksum code is: 11010110111110
Enter checksum code: 11010110111110
Enter generator: 10011
Data stream is valid
ubuntu@ubuntu-virtual-machine:~/java$ java CRC1
Enter message bits: 1101011011
Enter generator: 10011
The checksum code is: 11010110111110
Enter checksum code: 11010110111111
Enter generator: 10011
Data stream is invalid. CRC error occurred.
ubuntu@ubuntu-virtual-machine:~/java$
```

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

```
import java.util.Scanner;

public class ford

{

private int D[];

private int num_ver;

public static final int MAX_VALUE = 999;

public ford(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();

ford b = new ford (num_ver);

b.BellmanFordEvaluation(source, A);

scanner.close();

}

}
```

OUTPUT:

```
Enter the number of vertices

5

Enter the adjacency matrix

0 6 5 0 0

0 0 0 -1 0

0 -2 0 4 3

0 0 0 0 3

0 0 0 0 0
```

Enter the source vertex

1

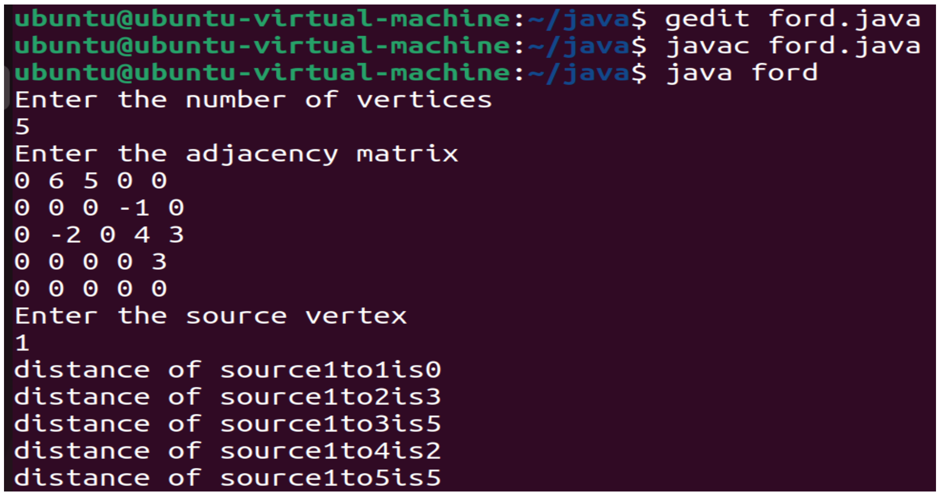
distance of source1to1is0

distance of source1to2is3

distance of source1to3is5

distance of source1to4is2

distance of source1to5is5



```
ubuntu@ubuntu-virtual-machine:~/java$ gedit ford.java
ubuntu@ubuntu-virtual-machine:~/java$ javac ford.java
ubuntu@ubuntu-virtual-machine:~/java$ java ford
Enter the number of vertices
5
Enter the adjacency matrix
0 6 5 0 0
0 0 0 -1 0
0 -2 0 4 3
0 0 0 0 3
0 0 0 0 0
Enter the source vertex
1
distance of source1to1is0
distance of source1to2is3
distance of source1to3is5
distance of source1to4is2
distance of source1to5is5
```

6. 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.

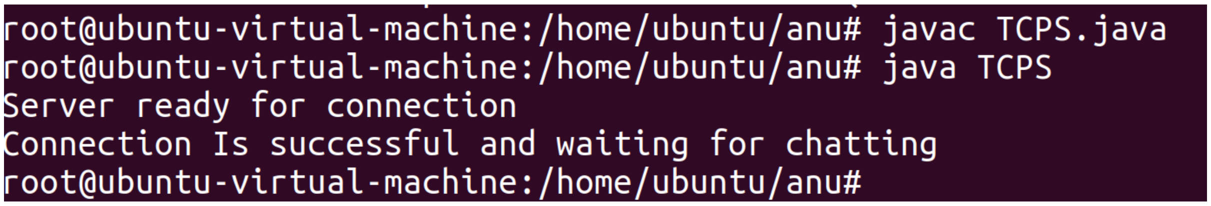
```
// TCP Server

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();

}

}


//TCP Client

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();

}

}
```

OUTPUT:

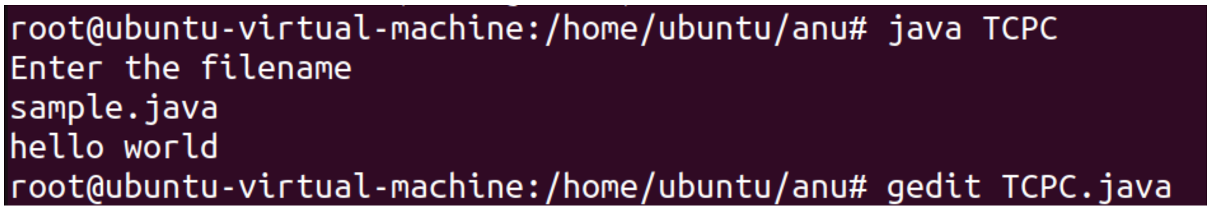
TCP SERVER:



```
root@ubuntu-virtual-machine:/home/ubuntu/anu# javac TCPS.java
root@ubuntu-virtual-machine:/home/ubuntu/anu# java TCPS
Server ready for connection
Connection Is successful and waiting for chatting
root@ubuntu-virtual-machine:/home/ubuntu/anu#
```

TCP CLIENT:

OPEN NEW TERMINAL :



```
root@ubuntu-virtual-machine:/home/ubuntu/anu# java TCPC
Enter the filename
sample.java
hello world
root@ubuntu-virtual-machine:/home/ubuntu/anu# gedit TCPC.java
```

7.Write a program on datagram socket for client/server to display the message on client side, typed at the server side.

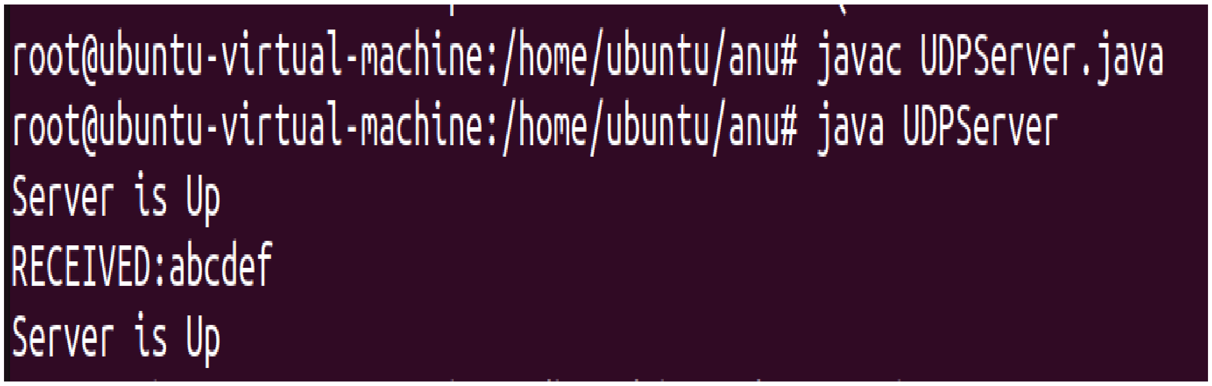
```
//UDP Sever

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);

}

}

}




//UDP Client

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();

}

}
```

OUTPUT:

UDP SERVER:

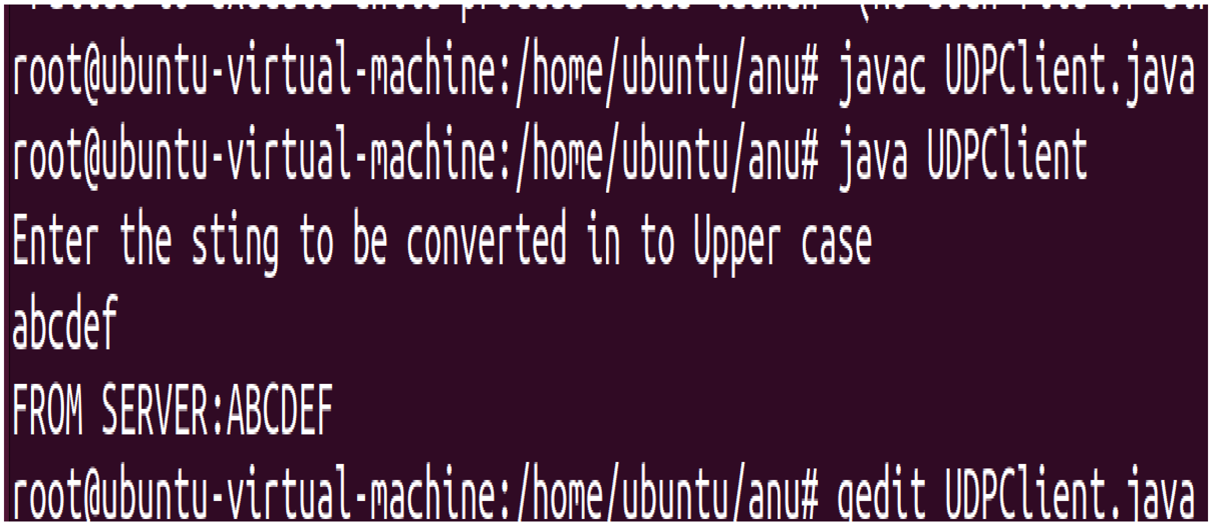
OPEN TERMINAL



```
root@ubuntu-virtual-machine:/home/ubuntu/anu# javac UDPServer.java
root@ubuntu-virtual-machine:/home/ubuntu/anu# java UDPServer
Server is Up
RECEIVED:abcdef
Server is Up
```

UDP CLIENT:

OPEN NEW TERMINAL



```
root@ubuntu-virtual-machine:/home/ubuntu/anu# javac UDPClient.java
root@ubuntu-virtual-machine:/home/ubuntu/anu# java UDPClient
Enter the sting to be converted in to Upper case
abcdef
FROM SERVER:ABCDEF
root@ubuntu-virtual-machine:/home/ubuntu/anu# gedit UDPClient.java
```

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

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
private BigInteger p,q,N,phi,e,d;
private int bitlength=1024;
private Random r;
public RSA()
{
r=new Random();
p=BigInteger.probablePrime(bitlength,r);
q=BigInteger.probablePrime(bitlength,r);
System.out.println("Prime number p is"+p);
System.out.println("prime number q is"+q);
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);
}
System.out.println("Public key is"+e);
d=e.modInverse(phi);
System.out.println("Private key is"+d);
}
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()));
byte[] encrypted=rsa.encrypt(testString.getBytes());
byte[] decrypted=rsa.decrypt(encrypted);
System.out.println("Dcrypting Bytes:"+bytesToString(decrypted));
System.out.println("Dcrypted 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)
```

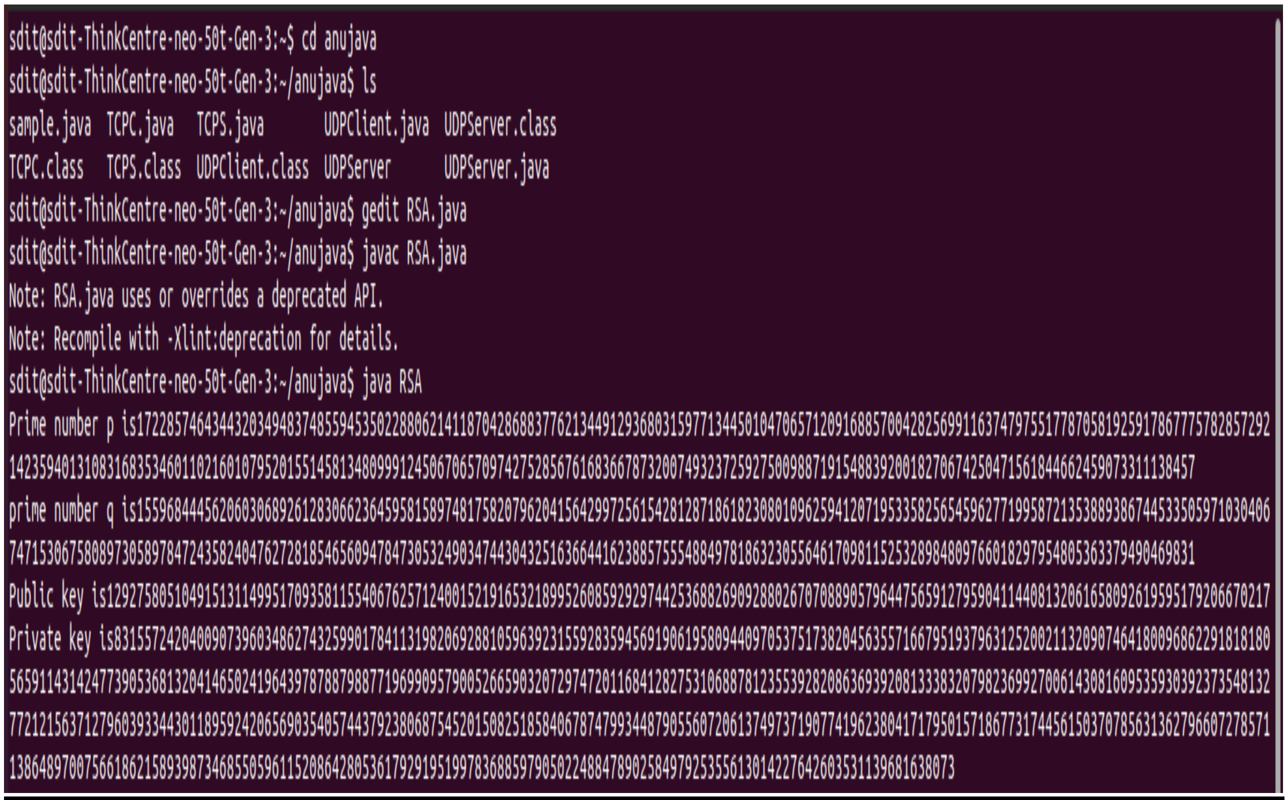
```
{
return(new BigInteger(message)).modPow(d,N).toByteArray();
}
}
```

OUTPUT:

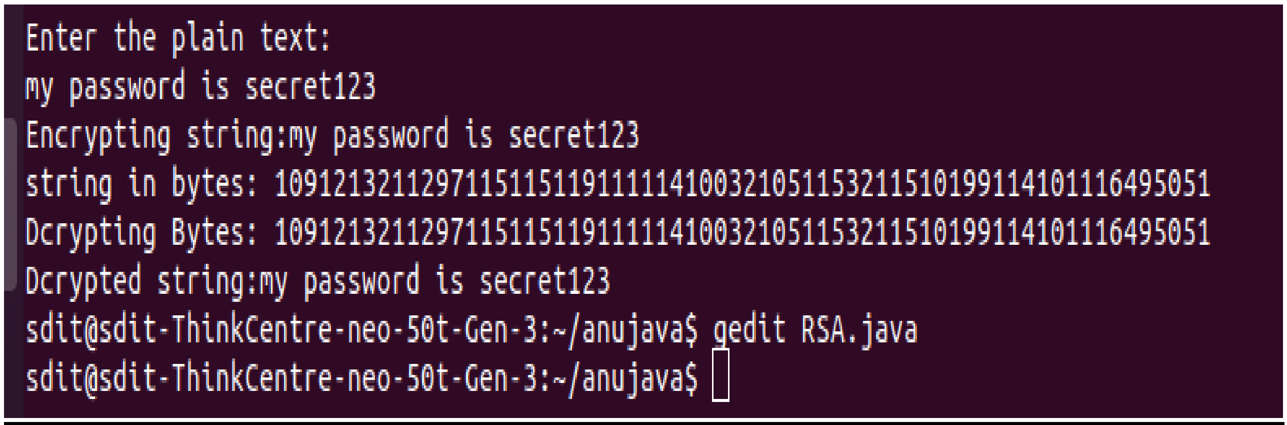
```
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ javac RSA.java
Note: RSA.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ java RSA
Prime number p
is17228574643443203494837485594535022880621411870428688377621344912
936803159771344501047065712091688570042825699116374797551778705819
259178677757828572921423594013108316835346011021601079520155145813
480999124506706570974275285676168366787320074932372592750098871915
4883920018270674250471561844662459073311138457
prime number q
is15596844456206030689261283066236459581589748175820796204156429972
561542812871861823080109625941207195335825654596277199587213538893
867445335059710304067471530675808973058978472435824047627281854656
094784730532490347443043251636644162388575554884978186323055646170
9811525328984809766018297954805363379490469831
Public key
is12927580510491513114995170935811554067625712400152191653218995260
859292974425368826909288026707088905796447565912795904114408132061
658092619595179206670217
Private key
is83155724204009073960348627432599017841131982069288105963923155928
359456919061958094409705375173820456355716679519379631252002113209
074641800968622918181805659114314247739053681320414650241964397878
879887719699095790052665903207297472011684128275310688781235539282
086369392081333832079823699270061430816095359303923735481327721215
637127960393344301189592420656903540574437923806875452015082518584
067874799344879055607206137497371907741962380417179501571867731744
561503707856313627966072785711386489700756618621589398734685505961
```

15208642805361792919519978368859790502248847890258497925355613014227642603531139681638073

Enter the plain text:
my password is secret123
Encrypting string:my password is secret123
string in bytes:
10912132112971151151191111141003210511532115101991141011116495051
Dcrypting Bytes:
10912132112971151151191111141003210511532115101991141011116495051
Dcrypted string:my password is secret123



```
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~$ cd anujava
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ ls
sample.java  TCPC.java   TCPS.java       UDPClient.java  UDPServer.class
TCPC.class   TCPS.class  UDPClient.class UDPServer       UDPServer.java
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ gedit RSA.java
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ javac RSA.java
Note: RSA.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ java RSA
Prime number p is17228574643443203494837485594535022880621411870428688377621344912936803159771344501047065712091688570042825699116374797551778705819259178677757828572921423594013108316835346011021601079520155145813480999124506706570974275285676168366787320074932372592750098871915488392001827067425047156184466245907331138457
prime number q is15596844456206030689261283066236459581589748175820796204156429972561542812871861823080109625941207195335825654596277199587213538893867445335059710304067471530675808973058978472435824047627281854656094784730532490347443043251636644162388575554884978186323055646170981152532898480976601829795480536337949046983l
Public key is12927580510491513114995170935811554067625712400152191653218995260859292974425368826909288026707088905796447565912795904114408132061658092619595179206670217
Private key is83155724204009073960348627432599017841131982069288105963923155928359456919061958094409705375173820456355716679519379631252002113209074641800968622918181805659114314247739053681320414650241964397878879887719699095790052665903207297472011684128275310688781235539282086369392081333832079823699270061430816095359303923735481327721215637127960393344301189592420656903540574437923806875452015082518584067874799344879055607206137497371907741962380417179501571867731744561503707856313627966072785711386489700756618621589398734685505961152086428053617929195199783688597905022488478902584979253556130142276426035311396816380 73
```

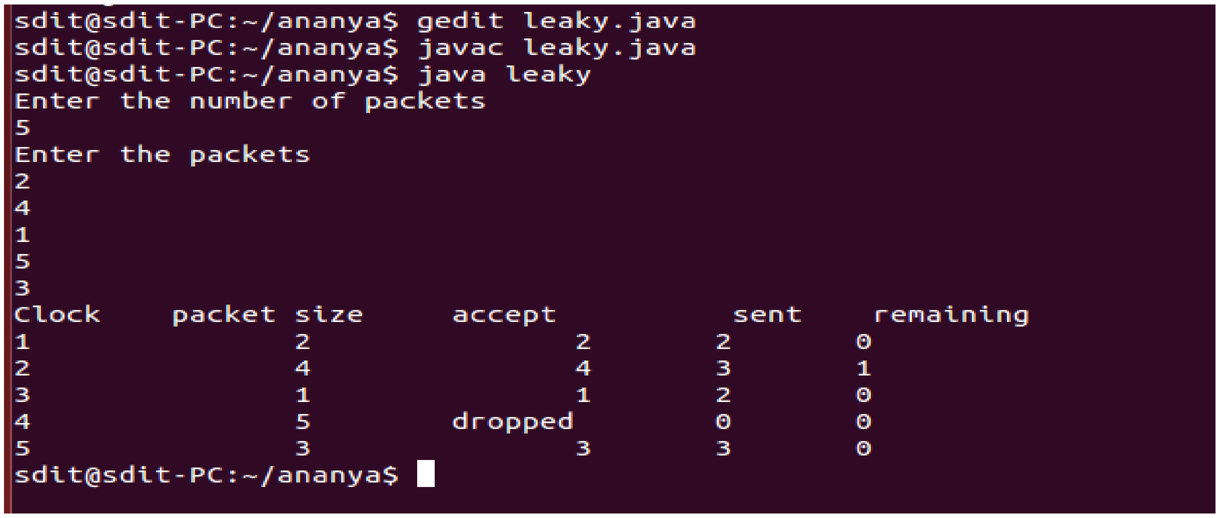


```
Enter the plain text:
my password is secret123
Encrypting string:my password is secret123
string in bytes: 10912132112971151151191111141003210511532115101991141011116495051
Dcrypting Bytes: 10912132112971151151191111141003210511532115101991141011116495051
Dcrypted string:my password is secret123
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$ gedit RSA.java
sdit@sdit-ThinkCentre-neo-50t-Gen-3:~/anujava$
```

9. Write a program for congestion control using leaky bucket algorithm.

```
import java.util.Scanner;
import java.lang.*;
public class leaky {
public static void main(String[] args)
{
int i;
int a[]=new int[20];
int buck_rem=0,buck_cap=4,rate=3,sent,recv;
Scanner in = new Scanner(System.in);
System.out.println("Enter the number of packets");
int n = in.nextInt();
System.out.println("Enter the packets");
for(i=1;i<=n;i++)
a[i]= in.nextInt();
System.out.println("Clock \t packet size \t accept \t sent \t remaining");
for(i=1;i<=n;i++)
{
if(a[i]!=0)
{
if(buck_rem+a[i]>buck_cap)
recv=-1;
else
{
recv=a[i];
buck_rem+=a[i];
}
}
else
recv=0;
if(buck_rem!=0)
{
if(buck_rem<rate)
{sent=buck_rem;
buck_rem=0;
}
else
{
sent=rate;
buck_rem=buck_rem-rate;
}
}
else
sent=0;
if(recv==-1)
System.out.println(+i+ "\t\t" +a[i]+ "\t dropped \t" +  sent +"\t" +buck_rem);
else
System.out.println(+i+ "\t\t" +a[i] +"\t\t" +recv +"\t" +sent + "\t" +buck_rem);
}
}
}
```

OUTPUT:



```
sdit@sdit-PC:~/ananya$ gedit leaky.java
sdit@sdit-PC:~/ananya$ javac leaky.java
sdit@sdit-PC:~/ananya$ java leaky
Enter the number of packets
5
Enter the packets
2
4
1
5
3
Clock    packet size     accept          sent     remaining
1                2              2        2        0
2                4              4        3        1
3                1              1        2        0
4                5       dropped         0        0
5                3              3        3        0
sdit@sdit-PC:~/ananya$
```