PART A

Program No. 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.

Program:

```
#Create Simulator
set ns [new Simulator]

#Open Trace file and NAM file
set ntrace [open prog1.tr w]
$ns trace-all $ntrace
set namfile [open prog1.nam w]
$ns namtrace-all $namfile

#Finish Procedure
proc Finish {} {
global ns ntrace namfile

#Dump all the trace data and close the files
$ns flush-trace
close $ntrace
close $namfile


#Execute the nam animation file
exec nam prog1.nam &

#Show the number of packets dropped
exec echo "The number of packet drops is " & exec grep -c "^d" prog1.tr &
exit 0
}
```

```
#Create 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

#Label the nodes
$n0 label "TCP Source"
$n2 label "Sink"

#Set the color
$ns color 1 blue

#Create Links between nodes
 #You need to modify the bandwidth to observe the variation in packet drop
 $ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail

#Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

#Set Queue Size
#You can modify the queue length as well to observe the variation in packet drop
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10

#Set up a Transport layer connection.
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $n2 $sink0
$ns connect $tcp0 $sink0

#Set up an Application layer Traffic
set cbr0 [new Application/Traffic/CBR]
$cbr0 set type_ CBR
$cbr0 set packetSize_ 100
$cbr0 set rate_ 1Mb
$cbr0 set random_ false
$cbr0 attach-agent $tcp0

$tcp0 set class_ 1

#Schedule Events
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "Finish"

#Run the Simulation
$ns run
```

Output Commands:

[root@localhost~] ns lab1.tcl

(OR)

Program:

```
set ns [new Simulator]

set tf [open lab1.tr w]
$ns trace-all $tf
```

```
set nf [open lab1.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns color 1 “red”
$ns color 2 “blue”
$n0 label “Source/udp0”
$n1 label “Source/udp1”
$n2 label “Router”
$n3 label “Destination/Null”

$ns duplex-link $n0 $n2 10Mb 300ms DropTail
$ns duplex-link $n1 $n2 10Mb 300ms DropTail
$ns duplex-link $n2 $n3 1Mb 300ms DropTail

$ns set queue-limit $n0 $n2 10
$ns set queue-limit $n1 $n2 10
$ns set queue-limit $n2 $n3 5
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set null [new Agent/Null]
$ns attach-agent $n3 $null
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
```

```
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

$udp0 set class_ 1
$udp1 set class_ 2

$ns connect $udp0 $null
$ns connect $udp1 $null

$cbr1 set packetSize_ 500Mb

$cbr1 set interval_ 0.005

proc finish { } {
global ns nf tf
$ns flush-trace
exec nam lab1.nam &
close $tf
close $nf
exit 0
}

$ns at 0.1 “$cbr0 start”
$ns at 0.1 “$cbr1 start”
$ns at 10.0 “finish”
$ns run
```

AWK file:

```
BEGIN{
#include<stdio.h>
count=0;
```

```
}
{
if($1==”d”)
count++
}
END{
printf(“The Total no of Packets Dropped due to Congestion:%d\n\n”,count);
}
```

Steps for execution:

- Open gedit and type program. Program name should have the extension “.tcl”
  [root@localhost~] gedit lab1.tcl
- Save the program
- Open gedit and type awk program. Program name should have the extension “.awk”
  [root@localhost~] gedit lab1.awk
- Save the program
- Run the simulation program
  [root@localhost~] ns lab1.tcl
- Here “ns” indicates network simulator. We get the topology shown in the snapshot
- Now press the play button in the simulation window and the simulation will begin
- After simulation is completed run awk file to see the output,
  [root@localhost~] awk -f lab1.awk lab1.tr
- To see the trace file contents, open the file as,
  [root@localhost~] gedit lab1.tr

Output:

The Total no of packets Dropped due to congestion: 456

Program No. 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.

Program:

```
#Create Simulator
set ns [new Simulator]

#Use colors to differentiate the traffic
$ns color 1 Blue
$ns color 2 Red

#Open trace and NAM trace file
 set ntrace [open prog3.tr w]
$ns trace-all $ntrace
set namfile [open prog3.nam w]
$ns namtrace-all $namfile

#Finish Procedure
proc Finish {} {
 global ns ntrace namfile

#Dump all trace data and close the file
$ns flush-trace close $ntrace close $namfile

#Execute the nam animation file
exec nam prog3.nam &

#Find the number of ping packets dropped
puts "The number of ping packets dropped are "
exec grep "^d" prog3.tr | cut -d " " -f 5 | grep -c "ping" & exit 0
}
```

```
#Create six nodes
for {set i 0} {$i < 6} {incr i} {
set n($i) [$ns node]
}

#Connect the nodes
for {set j 0} {$j < 5} {incr j} {
 $ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail
}

#Define the recv function for the class 'Agent/Ping'
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from $from with round trip time $rtt ms"
}

#Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]
$p0 set class_ 1
$ns attach-agent $n(0) $p0
set p1 [new Agent/Ping]
$p1 set class_ 1
$ns attach-agent $n(5) $p1
$ns connect $p0 $p1

#Set queue size and monitor the queue
#Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5
```

```
#Create Congestion
#Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply CBR traffic over TCP
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0

#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"

#Run the Simulation
$ns run
```

Output Commands:

[root@localhost~]#ns prgm2.tcl

(OR)

Program:

```
set ns [new Simulator]

set nf [open lab2.nam w]
$ns namtrace-all $nf

set nd [open lab2.tr w]
$ns trace-all $nd

proc finish { } {
global ns nf  nd
$ns flush-trace
close $nf
close $nd
exec nam lab2.nam &
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} {
$self instvar node_
puts “node [$node_id] received ping answer from \
$from with round-trip-time $rtt ms.”
}

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

$ns at 0.2 “$p1 send”
$ns at 0.4 “$p2 send”
$ns at 0.6 “$p3 send”
$ns at 1.0 “$p4 send”
$ns at 1.2 “$p5 send”
$ns at 1.4 “$p6 send”
$ns at 2.0 “finish”
$ns run
```

AWK file:

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

Output Commands:

[root@localhost~] ns lab2.tcl

[root@localhost~] awk -f prg2.awk lab2.tr

Output:

The Total no of packets dropped due to congestion:60

Program No. 3: Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
#Create Simulator
set ns [new Simulator]

#Use colors to differentiate the traffics
$ns color 1 Blue
$ns color 2 Red

#Open trace and NAM trace file
set ntrace [open prog5.tr w]
$ns trace-all $ntrace
set namfile [open prog5.nam w]
$ns namtrace-all $namfile

#Use some flat file to create congestion graph windows
set winFile0 [open WinFile0 w]
set winFile1 [open WinFile1 w]

#Finish Procedure
proc Finish {} {
#Dump all trace data and Close the files
global ns ntrace namfile
$ns flush-trace
close $ntrace
close $namfile

#Execute the NAM animation file
exec nam prog5.nam &

#Plot the Congestion Window graph using xgraph
```

```
exec xgraph WinFile0 WinFile1 &
exit 0
}

#Plot Window Procedure
proc PlotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]

set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}

 #Create 6 nodes
for {set i 0} {$i<6} {incr i} { set n($i) [$ns node]
}

#Create duplex links between the nodes
$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.6Mb 100ms DropTail

#Nodes n(3) , n(4) and n(5) are considered in a LAN
set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3
Channel]

#Orientation to the nodes
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
```

```
$ns duplex-link-op $n(2) $n(3) orient right

#Setup queue between n(2) and n(3) and monitor the queue
$ns queue-limit $n(2) $n(3) 20
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

 #Set error model on link n(2) to n(3)
set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

#Set up the TCP connection between n(0) and n(4)
set tcp0 [new Agent/TCP/Newreno]
$tcp0 set fid_ 1
$tcp0 set window_ 8000
$tcp0 set packetSize_ 552
$ns attach-agent $n(0) $tcp0
set sink0 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply FTP Application over TCP
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

 #Set up another TCP connection between n(5) and n(1)
 set tcp1 [new Agent/TCP/Newreno]
$tcp1 set fid_ 2
```

```
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
$ns attach-agent $n(5) $tcp1
set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(1) $sink1
$ns connect $tcp1 $sink1

#Apply FTP application over TCP
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

#Schedule Events
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "PlotWindow $tcp0 $winFile0"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "PlotWindow $tcp1 $winFile1"
$ns at 25.0 "$ftp0 stop"
$ns at 25.1 "$ftp1 stop"
$ns at 25.2 "Finish"

#Run the simulation
$ns run
```

Output Commands:

[root@localhost~]#ns prgm3.tcl

PART B

Program No. 7: Write a program for error detecting code using CRC-CCITT (16-bits)

```
import java.util.Scanner;
class CRC
{
        static String datastream;
        static String generator = "10001000000100001";
                public static void main(String args[])
                {
                        Scanner sc = new Scanner(System.in);
                        System.out.println("---At the Sender---\n Enter data stream:");
String datastream = sc.nextLine();
int datalen = datastream.length();
int genlen = generator.length();
int data[] = new int[datalen + genlen – 1];
int codeword = new int[datalen + genlen – 1];
int div[] = new int[generator.length()];
for(int i=0;i<datastream.length();i++)
                        data[i] = Integer.parseInt(datastream.charAt(i)+"");
for(int i=0;i<generator.length();i++)
                        div[i] = Integer.parseInt(generator.charAt(i)+"");
                        codeword = calculateCrc(data,div,datalen);
                        System.out.println("The CRC(Final Codeword) code is:");
for(int i=0;i<datastream.length();i++)
                        codeword[i] = Integer.parseInt(datastream.charAt(i)+"");
for(int i=0;i<data.length;i++)
                        System.out.print(codeword[i]);
                        System.out.println("\n");
                        System.out.print("---At the Receiver---\n Enter Received codeword:");
                        datastream = sc.nextLine();
```

```
                    data = new int[datastream.length() + generator.length() – 1];
                    for(int i=0;i<datastream.length();i++)
data[i] = Integer.parseInt(datastream.charAt(i)+"");
codeword = calculateCrc(data,div,datalen);
boolean valid = true;
for(int i=0;i<codeword.length;i++)
if(codeword[i]==1)
{
        valid = false;
        break;
}
if(valid==true)
System.out.println("Data stream is valid. No error occurred");
else
System.out.println("Data stream is invalid. CRC error occurred");
sc.close();
                }
                Public static int[] calculateCRC(int[] divrem, int[] divisor, int len)
                {
                    for(int i=0;i<len;i++)
                    {
                        if(divrem[i]==1)
                        for(int j=0;j<divisor.length;j++)
                        divrem[i+j]^=divisor[j];
                    }
                    return  divrem;
                }
}
```

Output Sample 1:

---At the Sender---

Enter data stream:

110101

The CRC(Final Codeword) code is:

110101011001101111 0110

---At the Receiver---

Enter Received codeword:

110101011001101111 10110

Data stream is valid. No error occurred

Output Sample 2:

---At the Sender---

Enter data stream:

110101

The CRC(Final Codeword) code is:

110101011001101111 0110

---At the Receiver---

Enter Received codeword:

100111100110010101 0101

Data stream is valid. No error occurred

Program No.8: Write a program to find the shortest path between vertices using bellman fort algorithm

```
import java.util.Scanner;
public class BellmanFord
{
private int dist[];
private int noofvert;
public static final int MAXVAL =999;
public BellmanFord(int noofvert)
{
this.noofvert=noofvert;
                dist=new int [noofvert+1];
}
        public void BellmanFordEval (int source, int adjmtx[][])
                {
                for (int node =1; node<=noofvert; node++)
                {
                        dist[node] = MAXVAL;
                }
                dist[source] =0;
for (int node=1; node<=noofvert-1; node++)
                {
                        for (int sn =1; sn<=noofvert;sn++)
                        {
                                for (int dn=1;dn<=noofvert;dn++)
                                        {
                                        if(adjmtx[sn][dn]!=MAXVAL)
                                                {
                                                        if(dist[dn]>dist[sn]+adjmtx[sn][dn])
                                                        dist[dn]=dist[sn]+adjmtx[sn][dn];
                                        }
```

```
}
}
}
System.out.println (“After (N-1)th Iteration”);
for (int v=1; v<=noofvert; v++)
{
System.out.println(“distance of source”+ source +”to”+ v +”is”+ dist[v]);
                }
                for(int sn=1;sn<=noofvert;sn++)
                {
                        for(int dn=1;dn<=noofvert;dn++)
                        {
                                if(adjmtx[sn][dn]!=MAXVAL)
                                {
                                        if(dist[dn]>dist[sn]+adjmtx[sn][dn])
                                        {
                                                dist[dn]=dist[sn]+adjmtx[sn][dn];
                                                System.out.println(“The  Graph contains
negative edge cycle”);
                                        }
                                }
                        }
                }
                System.out.println(“After Nth Iteration”);
                for(int v=1;v<=noofvert;v++)
                {
                        System.out.println(“distance of source”+ source + “to” + v + “is” +
dist[v]);
                }
        }
        public static void main(String[] args)
```

```
{
        int noofvert=0;
        int source;
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of vertices");
        noofvert=scanner.nextInt();
        int adjmtx[][]=new int[noofvert+1][noofvert+1];
        System.out.println("Enter the adjacency matrix");
        for(int sn=1;sn<=noofvert;sn++)
        {
                for(int dn=1;dn<=noofvert;dn++)
                {
                        adjmtx[sn][dn]=scanner.nextInt();
                        if(sn==dn)
                        {
                                adjmtx[sn][dn]=0;
                                continue;
                        }
                        if(adjmtx[sn][dn]==0)
                        {
                                adjmtx[sn][dn]=MAXVAL;
                        }
                }
        }
        System.out.println("Enter the source vertex");
        source=scanner.nextInt();
        BellmanFord bellmanford=new BellmanFord(noofvert);
        bellmanford.BellmanFordEval(source,adjmtrx);
        scanner.close();
}
```

}

Output Sample 1:

Enter the number of vertices

4

Enter the adjacency matrix

0 5 1 4

5 0 6 2

1 6 0 3

4 2 3 0

Enter the source vertex 1

After (N-1)th Iteration

distance of source 1 to 1 is 0

distance of source 1 to 2 is 5

distance of source 1 to 3 is 1

distance of source 1 to 4 is 4

After Nth Iteration

distance of source 1 to 1 is 0

distance of source 1 to 2 is 5

distance of source 1 to 3 is 1

distance of source 1 to 4 is 4

Output Sample 2:

Enter the number of vertices

6

Enter the adjacency matrix

0 4 1 999 999 999

999 0 999 999 1 2

999 999 0 3 999 999

999 999 999 0 999 999

-6 999 2 4 0 999

999 999 999 5 999 0

Enter the source vertex 1

After (N-1)th Iteration

distance of source 1 to 1 is -5

distance of source 1 to 2 is 0

distance of source 1 to 3 is -3

distance of source 1 to 4 is 0

distance of source 1 to 5 is 1

distance of source 1 to 6 is 2

After Nth Iteration

distance of source 1 to 1 is -6

distance of source 1 to 2 is -1

distance of source 1 to 3 is -4

distance of source 1 to 4 is -1

distance of source 1 to 5 is 0

distance of source 1 to 6 is 1

Program no.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.

Client Side program:

```
import java.io.*;
import java.net.*;
public class FileClient
{
public static void main (string [] args)
{
new FileClient();
}
        public FileClient()
        {
                BufferedReader bufReader=new BufferedReader(new
InputStreamReader(System.in));
                try
                {
                        System.out.println ("Enter IP address of the server:");
                        String saddr=bufReader.readLine();
                        Socket clientsocket=new Socket(saddr,8000);
                        System.out.println("Connecting to Server….");
                        DataInputStream input=new
DataInputStream(clientsocket.getInputStream ());
DataOutputStream output=DataOutputStream(clientsocket.getOutputStream());
System.out.println("Enter File Name:");
String Name=bufReader.readLine();
output.writeUTF(Name);
String EchoedFile=input.readUTF();
System.out.println("----------------------------------------");
Systen.out.println("Content of a File:\n\n"+EchoedFile);
System.out.println("----------------------------------------");
```

```
clientsocket.close();
                }
                catch(IOException ex)
                {
                        ex.printStackTrace();
                }
        }
}
```

Server side program:

```
import java.io.*;
import java.net.*;
public class FileServer
{
public static void main (string[] args)
{
new FileServer();
}
        public FileServer()
        {
                try
                {
                        ServerSocket serversocket=new ServerSocket(8000);
                        System.out.println (“Server Started....”);
                        System.out.println (“-----------------------------------------“);
                        Socket socket=serversocket.accept();
                        DataInputStream input=new
DataInputSteam(socket.getInputStream());
DataOutStream=new DataOutputStream(socket.getOutputStream());
String str=input.readUTF();
System.out.println(“Requested File Name:”+str);
System.out.println (“-----------------------------------------“);
```

```
try
{
InputSteam in =new FileInputStream(str);
BufferedReader reader=new bufferedReader (new InputStreamReader(in));
StringBuilder out=new StringBuilder();
String line;
System.out.println (“Reading Contents of the File...’);
System.out.println (“----------------------------------------“);
while (line=reader.readLine ())! =null)
{
out. append(line+”\n”);
}
String everything=out.toString();
System.out.println(“File Contents sent to  client...”);
System.out.println (“----------------------------------------“);
}
catch (Exception ex)
                        {
everything=”File Not Found!”);
}
output.writeUTF (everything);
}
catch (Exception ex)
{
ex.printStackTrace ();
}
        }
}
```

Note: Create two different files Client.java and Server.java. Follow the steps given:

1. Open a terminal run the server program and provide the filename to send.

2. Open the terminal run the client program and provide the IP address of the server.

3. Send any start bit to start sending file.

Output:

Server Side:

Server Started….

Requested File Name:abc.txt

Client Side:

Enter IP address of the server:

127.0.0.1

Connecting to Server….

Enter File Name:

abc.txt

Content of a File:

The content of the file will be displayed.

Program no.10: Write a program on datagram socket for client/server to display the messages on client side, typed at the server side

Program:

Client Side Program:

```
Import java.io*;
import java.net.*;
Class UDPCilent
{
public static void main (string args[]) throws Exception
{
BufferedReader inFormUser = new bufferedReader (new InputStreamReader(System.in));
System.out.println (“Enter the IP address of the Server :”);
String saddr = inFormUser.readLine();
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName(saddr);
byte[] receiveData;
byte[] sendData = new byte[200];
String sentence = ”Hello”;
sendData = sentence.getBytes ();
DatagramPacket sendPacket = new DatagramPacket(sentData, sendData.length, IPAddress,
9876) ;
clientSocket.send(sendPacket);
while(true)
{
receiveData = new byte[200];
DatagramPacket recivePacket = new Datagrampacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
string incomingData = new String(receivePacket.getData());
InetAddress SAddress = receivePacket.getAddress();
System.out.println(“FROM SERVER”+”(“+SAddress.toString()+”):+incomingData);
System.out.print\n (“-----------------------------------------------“);
```

```
}
}
}
```

Server side program:

```
import java.io.*;
import java.net.*;
class UDPServer
{
	public static void main(String args[]) throws Exception
	{
		DatagramSocket serverSocket = new DatagramSocket(9876);
		System.out.println("--------Server Started--------");
		BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
		byte[] receiveData = new byte[200];
		byte[] sendData;
		DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
		serverSocket.receive(receivePacket);
		InetAddress clientAddress = receivePacket.getAddress();
		int port = receivePacket.getPort();
		System.out.println("Client with IP
Address"+clientAddress.toString()+"connected...");
		System.out.println("----------------------------------------");
		System.out.println("Enter the message to send to client:");
		while(true)
		{
			String input = inFromUser.readLine();
			sendData = new byte[200];
			sendData = input.getBytes();
			DatagramPacket sendPacket = new DatagramPacket(SendData,
sendData.length, clientAddress, port);
```

```
                    serverSocket.send(sendPacket);
                }
            }
}
```

Note: Create two different files UDPClient.java and UDPServer.java.

Follow the following steps:

Open a terminal run the server program.

Open one more terminal run the client program, the sent message will be received.

Output:

Server Side:

--------Server Started--------

Client with IP Address/127.0.0.1 connected...

Enter the message to send to client:

Hello

Client with IP Address/127.0.0.1 connected...

Client Side:

Enter the IP address of the server:

Localhost

FROM SERVER(/127.0.0.1): Hello

Program no.11: Write a program for simple RSA algorithm to encrypt and decrypt the data

Program:

```
import java.math.Biglnteger;
import java.util.*;
Class RSA
{
public static void main(string args[])
{
Scanner ip = new Scanner (system.in);
int p, q, n, e=1, j;
int d=1, i1;
int pt[] = new int[10];
                int ct[] = new int[10];
int rt[] = new int[10];
int temp[] = new int[10];
String i= new String();
System.out.println (“Enter the two prime numbers :”);
p=ip.nextInt();
q=ip.nextInt();
System.out.println (“Enter the message to be sent”);
i=ip.next();
i1=i.length ();
n=p*q;
t1=p-1;
t2=q-1;
System.out.println(“\n-------------------------“);
System.out.println(“sender side:”);
while ((t1*t2)%e==0)
e++;
System.out.println(“Public Key(e)=”+e);
```

```
System.out.println (“----------");
for(j=0;j<i1;j++)
{
        pt[j]=(i.charAt(j))-96;
        System.out.println(“Plain Text=”+pt[j]);
        ct[j]=((int)Math.pow(pt[j],e))%n;
        System.out.println(“Cipher Text=”+ct[j]);
}
System.out.println(“\nTransmitted Message:”);
for(j=0;j<i1;j++)
{
temp[j]=ct[j]+96;
System.out.println((char)temp[j]);
}
System.out.println(“\n\n----------“);
System.out.println(“Receiver side:”);
while((d*e)%(t1*t2)!=1)
d++;
System.out.println(“Private Key(d)=”+d);
System.out.println(“----------“);
for(j=0;j<i1;j++)
{
System.out.println(“Cipher Text =”+ct[j]);
Biglnteger very_big_no=Biginteger.valueOf(ct[j]);
very_big_no=very_big_no.pow(d);
very_big_no=very_big_no.mod(BigInteger.valueOf(n));
rt[j]=very_big_on.intValue();
System.out.println(“Plain Text=”+rt[j]);
}
System.out.println(“\n----------“);
```

```
System.out.println(“Decrypted Message:”);
for(j=0;j<i1;j++)
{
rt[j]=rt[j]+96;
System.out.print((char))rt[j]);
}
        }
System.out.println(“\n--------------------------“);
ip.close();
}
}
```

Output

```
Enter the two prime numbers:
7        11
Enter the message to be sent
global
-------------------------------------
Sender Side:
Public Key(e)=7
-------------------------------------
Plain Text=7
Cipher Text=28
Plain Text=12
Cipher Text=12
Plain Text=15
Cipher Text=71
Plain Text=2
Cipher Text=51
Plain Text=1
Cipher Text=1
```

Plain Text=12

Cipher Text=12

Transmitted Message:

|1$?al

Receiver Side:

Private Key(d)=43

Cipher Text=28

Plain Text=7

Cipher Text=12

Plain Text=12

Cipher Text=71

Plain Text=15

Cipher Text=51

Plain Text=2

Cipher Text=1

Plain Text=1

Cipher Text=12

Plain Text=12

Decrypted Message:

Global

Program No.12: Write a program for congestion control using leaky bucket algorithm.

Program:

```
import java.util.scanner;
public class LeakyBucket
{
public static void main (string args[])
{
Scanner sc=new Scanner(System.in);
int incoming, outgoing, buck_size, n, time=1, store=0;
System.out.println (“Enter bucket size, outgoing rate and Number of Packets:”);
Buck_size=sc.nextInt();
n=sc.nextInt();
while(n!=0)
{
System.out.println(“Enter the incoming packet size at Time:”+(time++));
incoming=sc.nextlnt();
System.out.println(“Incoming packet size is “+incoming);
if(incoming<=(buck_size-store))
{
store+=incoming;
System.out.println(“Bucket buffer size is “+store+”out of “+buck_size);
}
else
{
int pktdrop=incoming-(buck_size-store);
Sytem.out.println(“Dropped”+pktdrop+”no of packets”);
System.out.println(“Bucket buffer size is 10 out of “+buck_size);
store=buck_size;
}
store=store-outgoing;
```

```
if(store<0)
{
store=0;
System.out.println (“Empty buffer”);
}
System.out.println(“After outgoing:”+store+”packets left out of”+buck_size+”in buffer\n”);
n--;
}
sc.close();
}
}
```

Output 1

```
Enter bucket size, outgoing rate and Number of Packets:
10      5       3
Enter the incoming packet size at Time: 1
16
Incoming packet size is 16
Dropped 6 no. of packets
Bucket buffer size is 10 out of 10
After outgoing: 5 packets left out of 10 in buffer
Enter the incoming packet size at Time: 2
6
Incoming packet size is 6
Dropped 1 no. of packets
Bucket buffer size is 10 out of 10
After outgoing: 5 packets left out of 10 in buffer
Enter the incoming packet size at Time: 3
4
Incoming packet size is 4
Bucket buffer size is 9 out of 10
```

After outgoing: 4 packets left out of 10 in buffer

Output 2

Enter bucket size, outgoing rate and Number of Packets:

8 2 4

Enter the incoming packet size at Time: 1

6

Incoming packet size is 6

Bucket buffer size is 6 out of 8

After outgoing: 4 packets left out of 8 in buffer

Enter the incoming packet size at Time: 2

10

Incoming packet size is 10

Dropped 6 no. of packets

Bucket buffer size is 10 out of 8

After outgoing: 6 packets left out of 8 in buffer

Enter the incoming packet size at Time: 3

12

Incoming packet size is 12

Dropped 10 no. of packets

Bucket buffer size is 10 out of 8

After outgoing: 6 packets left out of 8 in buffer

Enter the incoming packet size at Time: 4

12

Incoming packet size is 12

Dropped 10 no. of packets

Bucket buffer size is 10 out of 8

After outgoing: 6 packets left out of 8 in buffer