CAMBRIDGE

INSTITUTE OF TECHNOLOGY

NORTH CAMPUS

Off International Airport Road, Kundana, Bengaluru - 562110

PRACTICAL LAB MANUAL

Computer Networks

SUBJCT CODE:- BCS502
SEMESTER:- V
COURSE TYPE:-INTEGRATED

Prepared by:-

Dr Sridhar R, Prof Somshekhar

Professor, Assistant Professor

Department of Computer Science & Engineering

Sl.NO	**Experiments**
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.
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.
3	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4	Develop a program for error detecting code using CRC-CCITT (16- bits).
5	Develop a program to implement a sliding window protocol in the data link layer.
6	Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.
7	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.
8	Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.
9	Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10	Develop a program for congestion control using a leaky bucket algorithm.

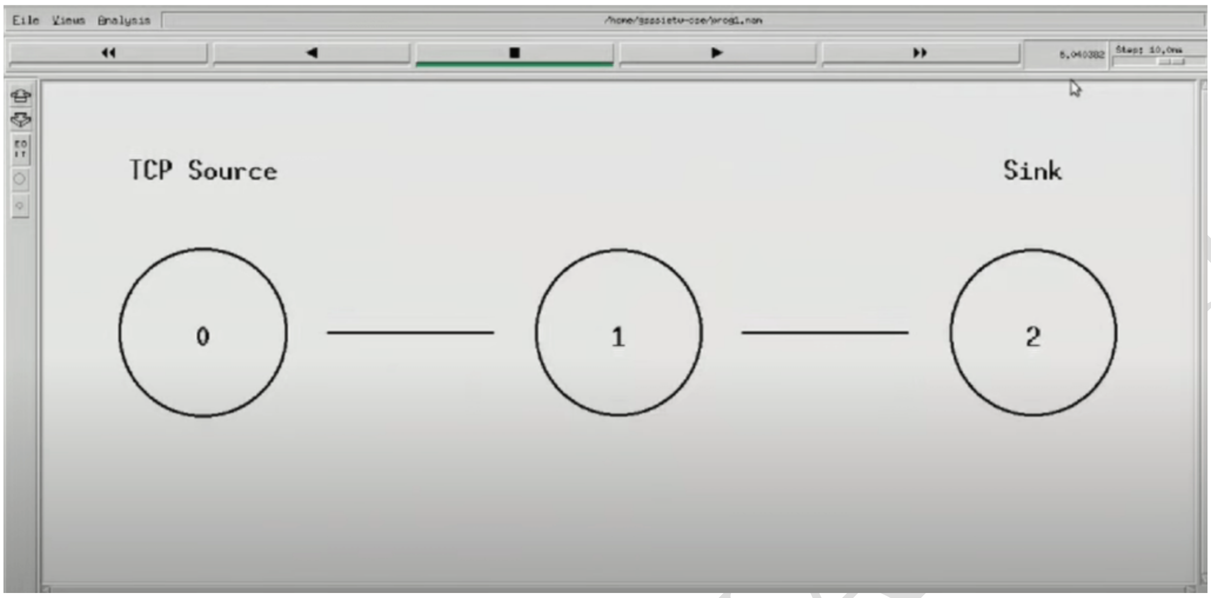
1)Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.

SOL)

```
#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 :



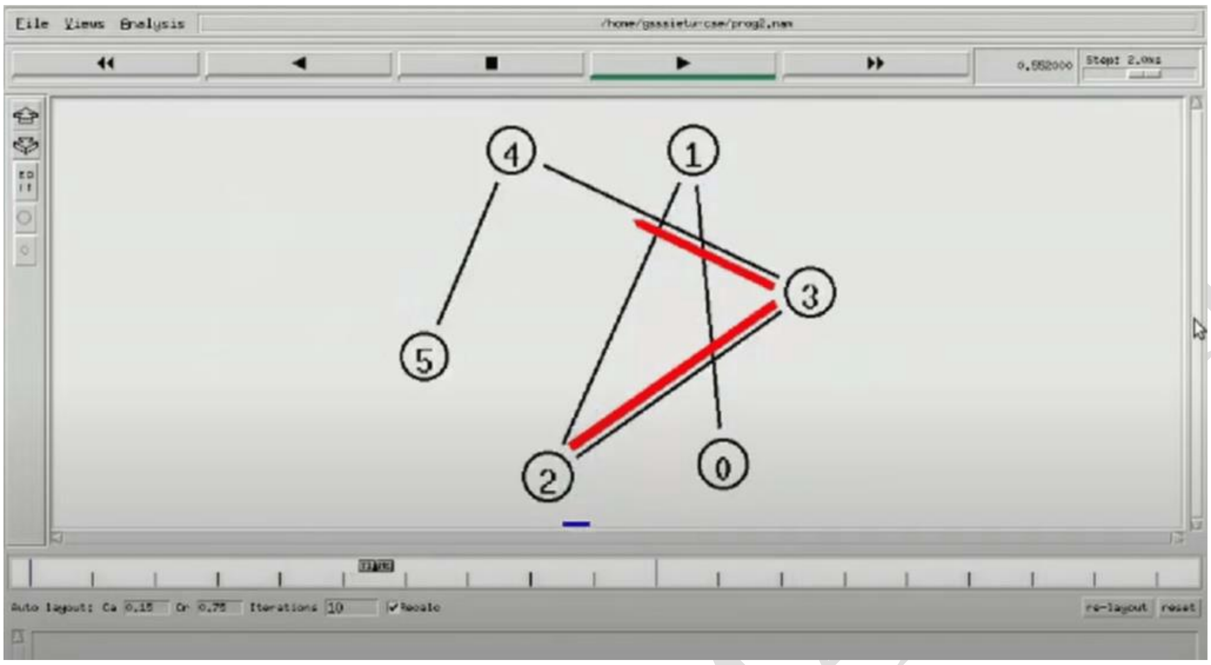
2) Implement transmission of ping messages/trace route over a network topology consisting of 6

nodes and find the number of packets dropped due to congestion.

Sol)

```
#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
```


File Views Analysis
/home/gssietu-cse/prog2.nam
0.552000
Step: 2.0ms
4
1
3
5
2
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Rescale
re-layout reset

3) Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion

window for different source / destination.

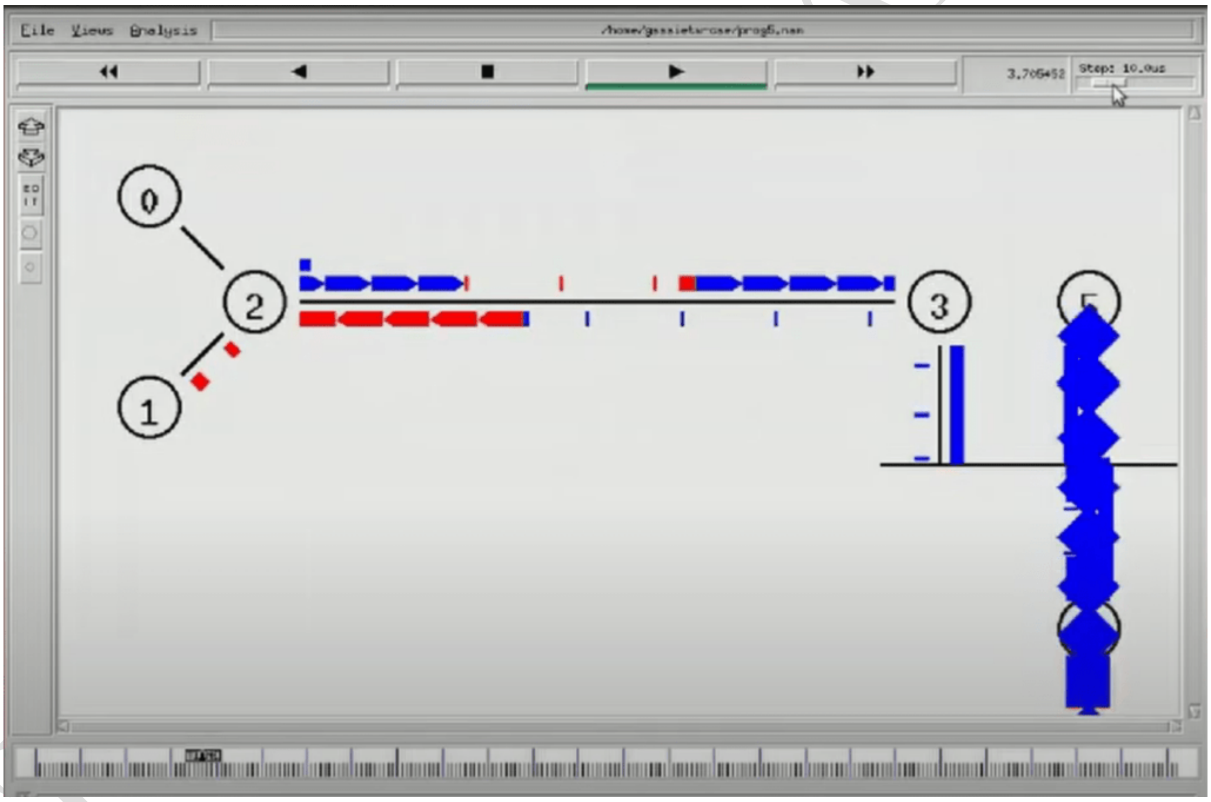
Sol)

```
#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
```

<u>OUTPUT</u> :



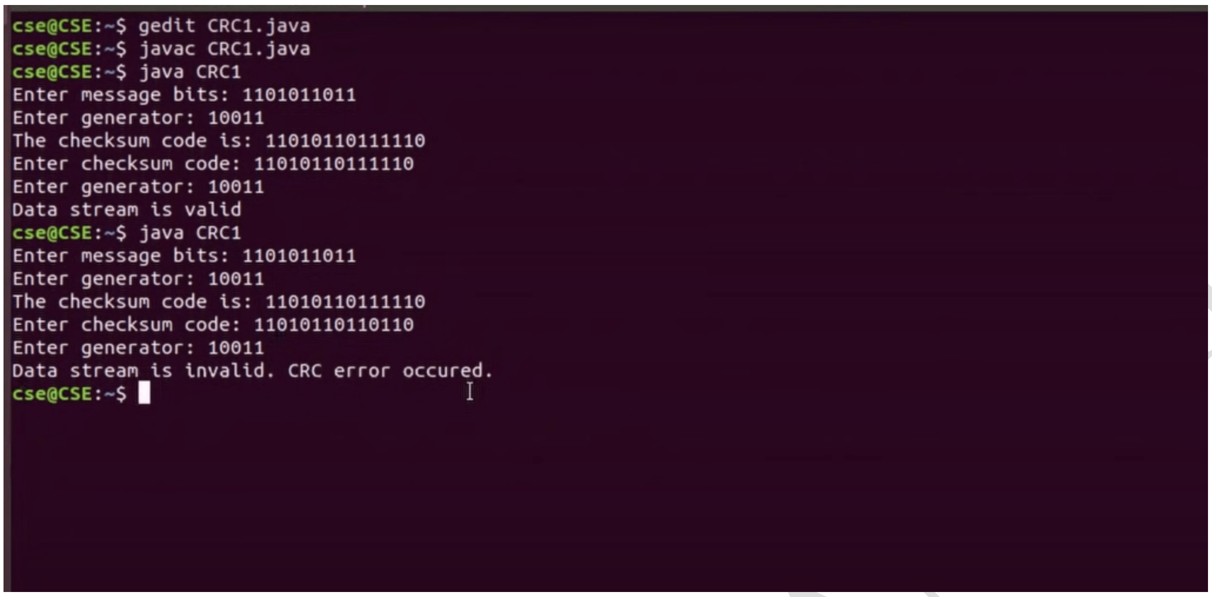
4) Develop a program for error detecting code using CRC-CCITT (16- bits).

Sol)

```
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.");
}

}
```

<u>OUTPUT</u> :



```
cse@CSE:~$ gedit CRC1.java
cse@CSE:~$ javac CRC1.java
cse@CSE:~$ 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
cse@CSE:~$ java CRC1
Enter message bits: 1101011011
Enter generator: 10011
The checksum code is: 11010110111110
Enter checksum code: 11010110110110
Enter generator: 10011
Data stream is invalid. CRC error occured.
cse@CSE:~$
```

5) Develop a program to implement a sliding window protocol in the data link layer. (C AND CPP)

Sol)

```
import java.util.Scanner;

public class SlidingWindowProtocol {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter window size: ");
        int w = scanner.nextInt();

        System.out.print("Enter number of frames to transmit: ");
        int f = scanner.nextInt();

        int[] frames = new int[f];

        System.out.println("Enter " + f + " frames: ");
        for (int i = 0; i < f; i++) {
            frames[i] = scanner.nextInt();
        }

        System.out.println("\nWith sliding window protocol, the frames will be sent
in the following manner:");
        System.out.println("After sending " + w + " frames at each stage, sender
waits for acknowledgment.");

        for (int i = 0; i < f; i++) {
            if ((i + 1) % w == 0) {
                System.out.println(frames[i]);
                System.out.println("Acknowledgment of above frames sent is received
by sender\n");
```

```
            } else {
                System.out.print(frames[i] + " ");
            }
        }

        if (f % w != 0) {
            System.out.println("\nAcknowledgment of above frames sent is received
by sender");
        }

        scanner.close();
    }
}
```

OUTPUT :

```
Enter window size: 3
Enter number of frames to transmit: 5
Enter 5 frames: 12 5 89 4 6
With sliding window protocol the frames will be sent in the following
manner (assuming nocorruption of frames)
After sending 3 frames at each stage sender waits for acknowledgement sent
by the receiver
12 5 89Acknowledgement of above frames sent is received by sender
4 6Acknowledgement of above frames sent is received by sender
```

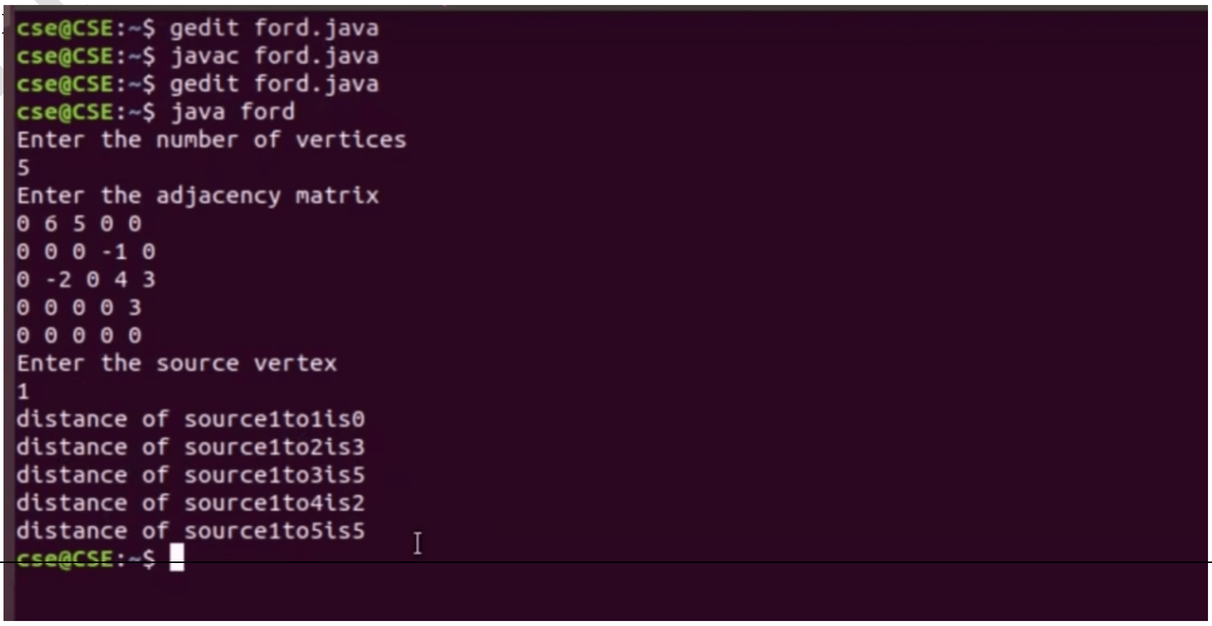
6) Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

Sol)

```
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 :



```
cse@CSE:~$ gedit ford.java
cse@CSE:~$ javac ford.java
cse@CSE:~$ gedit ford.java
cse@CSE:~$ 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
cse@CSE:~$
```

7) 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.

Sol)

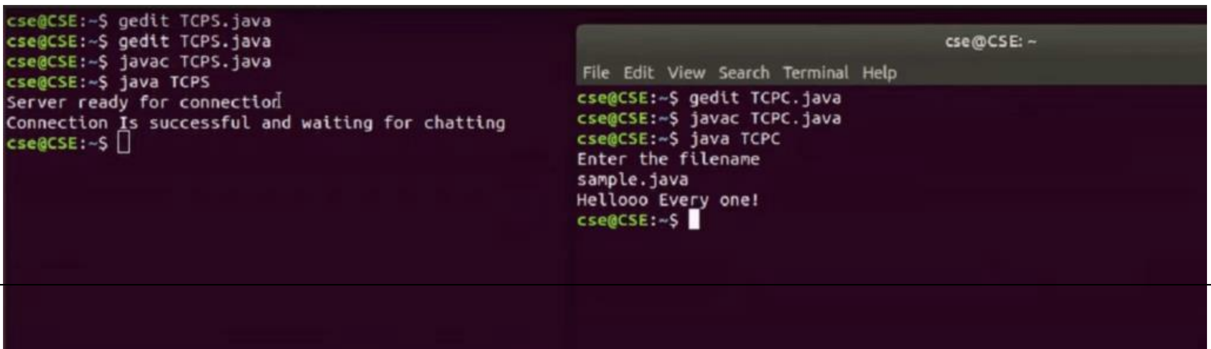
i) Program Code TCP SERVER

```
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();
}
}
```

ii) Program Code 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 :



```
cse@CSE:~$ gedit TCPS.java
cse@CSE:~$ gedit TCPS.java
cse@CSE:~$ javac TCPS.java
cse@CSE:~$ java TCPS
Server ready for connection
Connection Is successful and waiting for chatting
cse@CSE:~$
```

```
cse@CSE:~$ gedit TCPC.java
cse@CSE:~$ javac TCPC.java
cse@CSE:~$ java TCPC
Enter the filename
sample.java
Hellooo Every one!
cse@CSE:~$
```

8) Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

Sol)

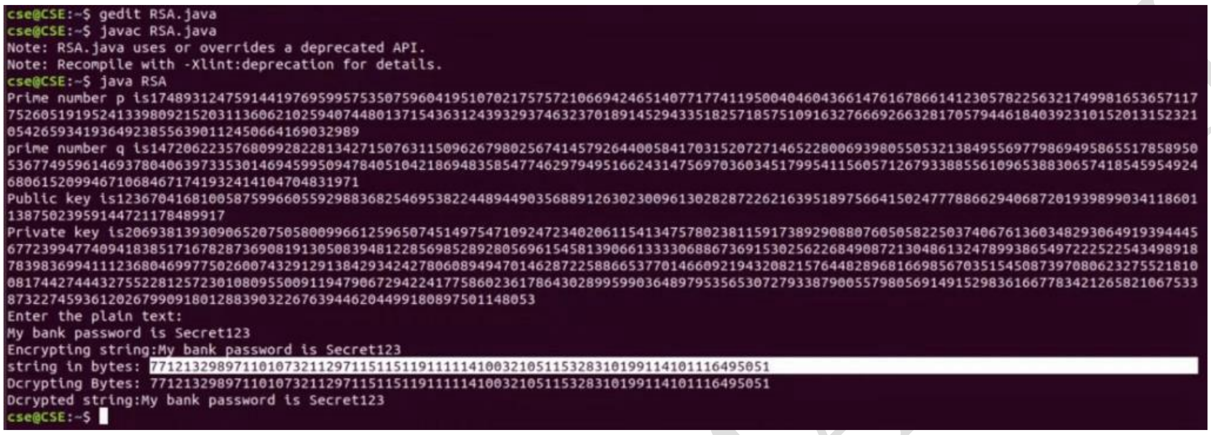
<u>i) Program code UDP SERVER :</u>

```
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);
}
}
}
```

ii) Program Code 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:



```
cse@CSE:~$ gedit RSA.java
cse@CSE:~$ javac RSA.java
Note: RSA.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
cse@CSE:~$ java RSA
Prime number p is17489312475914419769599575350759604195107021757572106694246514077177411950040460436614761678661412305782256321749981653657117
7526051919524133980921520311360621025940744801371543631243932937463237018914529433518257185751091632766692663281705794461840392310152013152321
054265934193649238556390112450664169032989
prime number q is14720622357680992822813427150763115096267980256741457926440058417031520727146522800693980550532138495569779869495865517858950
5367749596146937804063973353014694599509478405104218694835854774629794951662431475697036034517995411560571267933885561096538830657418545954924
680615209946710684671741932414104704831971
Public key is123670416810058759966055929883682546953822448944903568891263023009613028287226216395189756641502477788662940687201939899034118601
13875023959144721178489917
Private key is20693813930906520750580099661259650745149754710924723402061154134757802381159173892908807605058225037406761360348293064919394445
6772399477409418385171678287369081913050839481228569852892805696154581390661333306886736915302562268490872130486132478993865497222522543498918
7839836994111236804699775026007432912913842934242780608949470146287225886653770146609219432082157644828968166985670351545087397080623275521810
0817442744432755228125723010809550091194790672942241775860236178643028995990364897953565307279338790055798056914915298361667783421265821067533
873227459361202679909180128839032267639446204499180897501148053
Enter the plain text:
My bank password is Secret123
Encrypting string:My bank password is Secret123
string in bytes: 77121329897110107321129711511511911111410032105115328310199114101116495051
Dcrypting Bytes: 77121329897110107321129711511511911111410032105115328310199114101116495051
Dcrypted string:My bank password is Secret123
cse@CSE:~$
```

9) Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

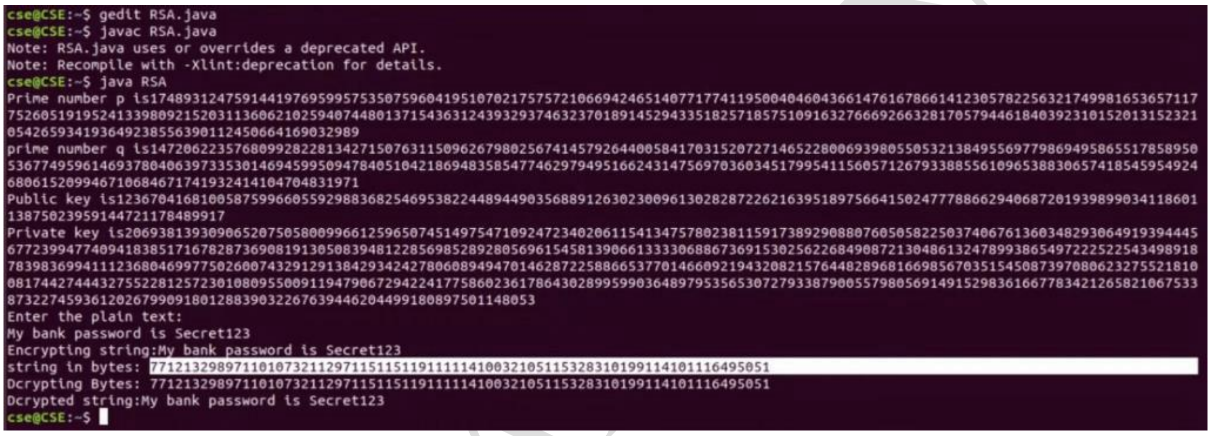
Sol)

```
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 :



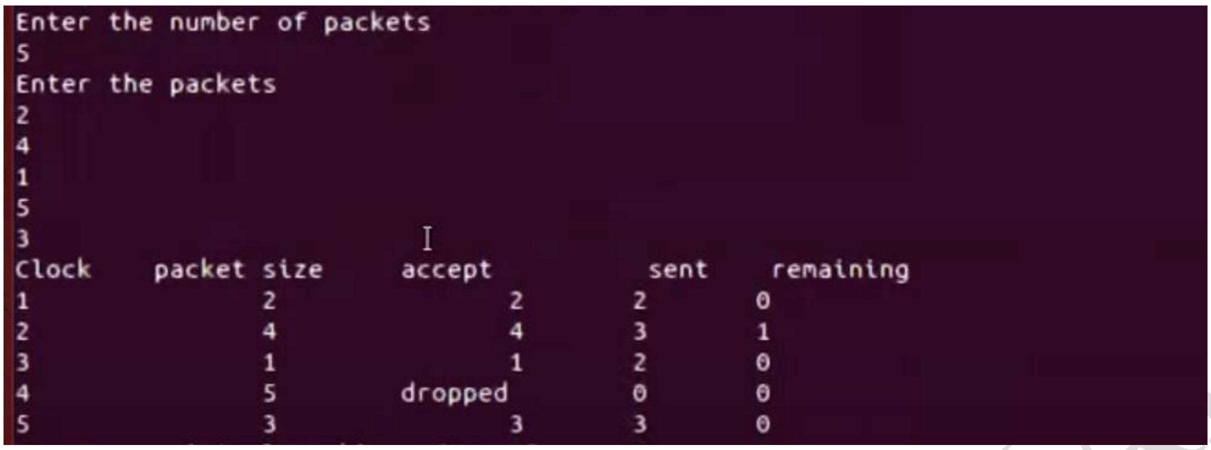
```
cse@CSE:~$ gedit RSA.java
cse@CSE:~$ javac RSA.java
Note: RSA.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
cse@CSE:~$ java RSA
Prime number p is17489312475914419769599575350759604195107021757572106694246514077177411950040460436614761678661412305782256321749981653657117
7526051919524133980921520311360621025940744801371543631243932937463237018914529433518257185751091632766692663281705794461840392310152013152321
054265934193649238556390112450664169032989
prime number q is1472062235768099282281342715076311509626798025674145792644005841703152072714652280069398055053213849556977986949586551785895O
5367749596146937804063973353014694599509478405104218694835854774629794951662431475697036034517995411560571267933885561096538830657418545954924
680615209946710684671741932414104704831971
Public key is1236704168100587599660559298836825469538224489449035688912630230096130282872262163951897566415024777886629406872019398990341186O1
13875023959144721178489917
Private key is2069381393090652075058009966125965074514975471092472340206115413475780238115917389290880760505822503740676136034829306491939444 5
6772399477409418385171678287369081913050839481228569852892805696154581390661333306886736915302562268490872130486132478993865497222522543498918
7839836994111236804699775026007432912913842934242780608949470146287225886653770146609219432082157644828968166985670351545087397080623275521810
0817442744432755228125723010809550091194790672942241775860236178643028995990364897953565307279338790055798056914915298361667783421265821067533
873227459361202679909180128839032267639446204499180897501148053
Enter the plain text:
My bank password is Secret123
Encrypting string:My bank password is Secret123
string in bytes: 77121329897110107321129711511511911111410032105115328310199114101116495051
Dcrypting Bytes: 77121329897110107321129711511511911111410032105115328310199114101116495051
Dcrypted string:My bank password is Secret123
cse@CSE:~$
```

10) Develop a program for congestion control using a leaky bucket algorithm

Sol)

```
import java.util.Scanner;
import java.lang.*;
public class lab7 {
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:


```
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
```