```
#Program 1

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]

$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 0.5Mb 300ms DropTail

$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
$ns queue-limit $n2 $n3 3

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

$udp0 set class_ 1
$ns color 1 Blue
$udp1 set class_ 2
$ns color 2 Red

set null [new Agent/Null]
$ns attach-agent $n3 $null

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

set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

$cbr1 set packetSize_ 500Mb
$cbr1 set interval_ 0.005

proc finish { } {
        global ns tf nf
        $ns flush-trace
        close $tf
```

```
        close $nf
        exec nam lab1.nam &
        exit 0
}

$ns at 0.1 "$cbr0 start"
$ns at 0.1 "$cbr1 start"
$ns at 10.0 "finish"
$ns run

#awk

BEGIN {
        count=0
}
{
        if($1=="d")
                count++
}
END {
        printf("Number of packets dropped is = %d\n",count);
}


#Program 2

set ns [new Simulator]
set tf [open lab2.tr w]
$ns trace-all $tf
set nf [open lab2.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n0 label "ping0"
$n1 label "ping1"
$n2 label "ping2"
$n3 label "ping3"
$n4 label "ping4"
$n5 label "router"

$ns duplex-link $n0 $n5 1Mb 10ms DropTail
$ns duplex-link $n1 $n5 1Mb 10ms DropTail
$ns duplex-link $n2 $n5 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns queue-limit $n0 $n5 5
$ns queue-limit $n1 $n5 5
```

```
$ns queue-limit $n2 $n5 2
$ns queue-limit $n3 $n5 5
$ns queue-limit $n4 $n5 2

set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
set p1 [new Agent/Ping]
$ns attach-agent $n1 $p1
set p2 [new Agent/Ping]
$ns attach-agent $n2 $p2
set p3 [new Agent/Ping]
$ns attach-agent $n3 $p3
set p4 [new Agent/Ping]
$ns attach-agent $n4 $p4

$p1 set class_ 1
$ns color 1 Red
$p2 set class_ 2
$ns color 2 Blue
$p3 set class_ 3
$ns color 3 Green
$p4 set class_ 4
$ns color 4 Yellow

$ns connect $p2 $p4
$ns connect $p3 $p4

#define ‘recv’ function for 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"
}

proc sendPingPacket {} {
        global ns p2 p3
        set time 0.001
        set now [$ns now]
        $ns at [expr $now + $time] "$p2 send"
        $ns at [expr $now + $time] "$p3 send"
        $ns at [expr $now + $time] "sendPingPacket"
}

proc finish {} {
        global ns tf nf
        $ns flush-trace
        close $tf
        close $nf
        exec nam lab2.nam &
        exit 0
}

$ns at 0.1 "sendPingPacket"
$ns at 2.0 "finish"
```

```
$ns run

#awk

BEGIN {
        count=0
}
{
        if($1=="d")
                count++
}
END {
        printf("Number of packets dropped is = %d\n",count);
}


#Program 3

set ns [new Simulator]
set tf [open lab3.tr w]
$ns trace-all $tf
set nf [open lab3.nam w]
$ns namtrace-all $nf
set winfile [open winfile w]

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 shape box

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.5Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.5Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 100ms LL Queue/DropTail MAC/802_3]

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n3 $n2 orient left
$ns simplex-link-op $n2 $n3 orient right

$ns queue-limit $n2 $n3 20

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1

$tcp0 set class_ 1
$ns color 1 Blue
$tcp1 set class_ 2
```

```
$ns color 2 Red

$tcp0 set packetSize_ 552
$tcp1 set packetSize_ 552

set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

$ns connect $tcp $sink
$ns connect $tcp1 $sink1

set ftp [new Application/FTP]
$ftp attach-agent $tcp0

set telnet [new Application/Telnet]
$telnet attach-agent $tcp1

set outfile1 [open congestion1.xg w]
puts $outfile1 "TitleText: Congestion Window-- Source _tcp0"
puts $outfile1 "xUnitText: Simulation Time(Secs)"
puts $outfile1 "yUnitText: Congestion WindowSize"

set outfile2 [open congestion2.xg w]
puts $outfile2 "TitleText: Congestion Window-- Source _tcp1"
puts $outfile2 "xUnitText: Simulation Time(Secs)"
puts $outfile2 "yUnitText: Congestion WindowSize"

proc plotWindow {tcpSource outfile} {
        global ns
        set time 0.1
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        puts $outfile "$now $cwnd"
        $ns at [expr $now + $time] "plotWindow $tcpSource $outfile"
}

proc finish { } {
        global ns tf nf
        $ns flush-trace
        close $tf
        close $nf
        exec nam lab3.nam &
        exit 0
}

$ns at 0.0 "plotWindow $tcp0 $winfile"
$ns at 0.0 "plotWindow $tcp0 $outfile1"
$ns at 0.0 "plotWindow $tcp1 $outfile2"
$ns at 0.1 "$ftp start"
$ns at 0.2 "$telnet start"
$ns at 49.0 "$ftp stop"
$ns at 49.1 "$telnet stop"
```

```
$ns at 50.0 "finish"
$ns run


#Program 4

set ns [new Simulator]
set tf [open lab4.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open lab4.nam w]
$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV \
                                -llType LL \
                                -macType Mac/802_11 \
                                -ifqType Queue/DropTail \
                                -ifqLen 20 \
                                -phyType Phy/WirelessPhy \
                                -channelType Channel/WirelessChannel \
                                -propType Propagation/TwoRayGround \
                                -antType Antenna/OmniAntenna \
                                -topoInstance $topo \
                                -agentTrace ON \
                                -routerTrace ON

create-god 4

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

$n0 label "tcp0"
$n1 label "sink0"
$n2 label "bs1"
$n3 label "bs2"

$n0 set X_ 110
$n0 set Y_ 500
$n0 set Z_ 0
$n1 set X_ 700
$n1 set Y_ 500
$n1 set Z_ 0
$n2 set X_ 300
$n2 set Y_ 500
$n2 set Z_ 0
$n3 set X_ 510
$n3 set Y_ 500
$n3 set Z_ 0

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
```

```
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 0.3 "$n0 setdest 100 500 10"
$ns at 0.3 "$n1 setdest 700 500 10"
$ns at 0.3 "$n2 setdest 300 500 10"
$ns at 0.3 "$n3 setdest 500 500 10"
$ns at 100.0 "$n0 setdest 200 700 5"
$ns at 100.0 "$n1 setdest 600 400 5"
$ns at 200.0 "$n0 setdest 100 800 5"
$ns at 200.0 "$n1 setdest 700 300 5"
$ns at 300.0 "$n0 setdest 100 400 5"
$ns at 300.0 "$n1 setdest 600 700 5"

proc finish {} {
        global ns tf nf
        $ns flush-trace
        close $tf
        close $nf
        exec nam lab4.nam &
        exit 0
}

$ns at 0.5 "$ftp start"
$ns at 400 "finish"
$ns run

#awk

BEGIN {
     PktsSent=0;
     PktsRcvd=0;
     PktsAtRTR=0;
}
{
     if(($1=="s")&&($4=="RTR")&&($7=="tcp")) PktsAtRTR++;
     if(($1=="s")&&($4=="AGT")&&($7=="tcp")) PktsSent++;
     if(($1=="r")&&($4=="AGT")&&($7=="tcp")) PktsRcvd++;
}
END {
     print " Number of Packets Sent :" PktsSent
     print " Number of Packets Received :" PktsRcvd
     print " Packet Delivery Ratio :" PktsRcvd/PktsSent*100
     print " Routing Load :" PktsAtRTR/PktsRcvd
}


#Program 5
```

```
set ns [new Simulator]
set tf [open lab5.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]

$ns duplex-link $n0 $n1 3Mb 50ms DropTail
$ns duplex-link $n1 $n2 10Kb 500ms RED
$ns duplex-link $n2 $n3 10Kb 500ms RED
$ns duplex-link $n3 $n4 3Mb 10ms DropTail

$ns queue-limit $n2 $n1 10
$ns queue-limit $n2 $n3 10

Queue/RED set adaptive_ 1
Queue/RED set thresh_ 30
Queue/RED set maxthresh_ 0
Agent/TCP set window_ 30

$ns insert-delayer $n2 $n1 [new Delayer]
$ns insert-delayer $n2 $n3 [new Delayer]

set tcp [new Agent/TCP/Sack1]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/Sack1]
$ns attach-agent $n4 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

proc stop {} {
        global n0 n1
        set wrap 100
        set sid [$n0 id]
        set did [$n1 id]
        set a "lab5.tr"
        set GETRC "/root/ns-allinone-2.35/ns-2.35/bin/getrc"
        set RAW2XG "/root/ns-allinone-2.35/ns-2.35/bin/raw2xg"

        exec $GETRC -s $sid -d $did -f 0 lab5.tr | \
                $RAW2XG -s 0.01 -m $wrap -r > plot5.xgr
        exec $GETRC -s $did -d $sid -f 0 lab5.tr | \
                $RAW2XG -a -s 0.01 -m $wrap >> plot5.xgr
        exec xgraph -x time -y packets plot5.xgr &
        exit 0
}

$ns at 0.8 "$ftp start"
```

```
$ns at 100 "stop"
$ns run


#Program 6

set ns [new Simulator]
set tf [open lab6.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]

$ns duplex-link $n0 $n1 3Mb 50ms DropTail
$ns duplex-link $n1 $n2 40Kb 200ms RED
$ns duplex-link $n2 $n3 40Kb 200ms RED
$ns duplex-link $n3 $n4 3Mb 10ms DropTail

$ns queue-limit $n2 $n1 20
$ns queue-limit $n2 $n3 20

$ns insert-delayer $n2 $n1 [new Delayer]
$ns insert-delayer $n2 $n3 [new Delayer]

Queue/RED set adaptive_ 1
Queue/RED set thresh_ 30
Queue/RED set maxthresh_ 0
Agent/TCP set window_ 30

set tcp [new Agent/TCP/Sack1]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/Sack1]
$ns attach-agent $n4 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

proc stop {} {
        global n0 n1
        set wrap 100
        set sid [$n0 id]
        set did [$n1 id]
        set a "lab6.tr"
        set GETRC "/root/ns-allinone-2.35/ns-2.35/bin/getrc"
        set RAW2XG "/root/ns-allinone-2.35/ns-2.35/bin/raw2xg"

        exec $GETRC -s $sid -d $did -f 0 lab6.tr | \
                $RAW2XG -s 0.01 -m $wrap -r > plot6.xgr
        exec $GETRC -s $did -d $sid -f 0 lab6.tr | \
```

```
                $RAW2XG -a -s 0.01 -m $wrap >> plot6.xgr
        exec xgraph -x time -y packets plot6.xgr &
        exit 0
}

$ns at 0.8 "$ftp start"
$ns at 100 "stop"
$ns run


#Program 7

import java.util.*;

public class lab7
{
        public static int n;
        public static void main(String[] args)
        {
                Scanner input = new Scanner(System.in);
                String copy, rec, code, zero = "0000000000000000";
                System.out.println("enter the dataword to be sent");
                code = input.nextLine();

                n = code.length();
                copy = code;
                code += zero;
                code = divide(code);
                System.out.println("dataword = " + copy);
                copy += code;

                System.out.print("CRC = ");
                System.out.println(code);
                System.out.println("transmitted frame is = " + copy);
                System.out.println("enter received data:");
                rec = input.nextLine();

                if(zero.equals(divide(rec)))
                        System.out.println("correct bits received");
                else
                        System.out.println("received frame contains one or more
error");
        }

        public static String divide(String s)
        {
                String div="10001000000100001";
                for(int i = 0; i < n; i++)
                        if(s.charAt(i) == '1')
                                for(int j = 0; j < 17; j++)
                                        if(s.charAt(i+j) != div.charAt(j))
                                                s = s.substring(0, i+j) + "1" +
s.substring(i+j+1);
                                        else
```

```
                                                s = s.substring(0, i+j) + "0" +
s.substring(i+j+1);
                return s.substring(n);
        }
}


#Program 8

import java.util.*;

public class lab8
{
        public static final int MAX = 999;

        public static void bellman(int source, int[][] matrix, int n)
        {
                int[] cost = new int[n+1];
                for(int node = 1; node <= n; node++)
                        cost[node] = MAX;
                cost[source] = 0;
                for(int node = 1; node <= n-1; node++)
                        for(int src = 1; src <= n; src++)
                                for(int dest = 1; dest <= n; dest++)
                                {
                                        int x = matrix[src][dest];
                                        if(matrix[src][dest] != MAX &&
cost[dest] > cost[src] + x)
                                                cost[dest] = cost[src] + x;
                                }
                for(int src = 1; src <= n; src++)
                        for(int dest = 1; dest <= n; dest++)
                        {
                                int x = matrix[src][dest];
                                if(x != MAX && cost[dest] > cost[src] + x)
                                        System.out.println("The graph contains
negative edge cycle");
                        }
                System.out.println("Routing Table for Router " + source + "
is");
                System.out.println("Destination\tDistance\t");
                for(int i = 1; i <= n; i++)
                        System.out.println(i + "\t\t\t" + cost[i]);
        }

        public static void main(String args[])
        {
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter the number of vertices");
                int n = scan.nextInt();
                int matrix[][] = new int[n+1][n+1];
                System.out.println("Enter the adjacency matrix");
                for(int src = 1; src <= n; src++)
                        for(int dest = 1;dest <= n; dest++)
```

```
                        {
                                int x = scan.nextInt();
                                if(src != dest && x == 0)
                                        x = MAX;
                                matrix[src][dest] = x;
                        }
                for(int i = 1; i <= n; i++)
                        bellman(i, matrix, n);
        }
}


#Program 9

#Client

import java.util.*;
import java.net.*;
import java.io.*;

public class lab9client
{
        public static void main(String args[])
        {
                try
                {
                        Scanner scan = new Scanner(System.in);
                        Socket s = new Socket("localhost",998);
                        DataInputStream dis = new
DataInputStream(s.getInputStream());
                        DataOutputStream dos = new DataOutputStream
(s.getOutputStream());
                        dos.writeUTF("connected to 127.0.0.1 \n");
                        System.out.println(dis.readUTF());
                        System.out.println("\n enter the full path of the file
to be displayed");
                        String path = scan.nextLine();
                        dos.writeUTF(path);
                        System.out.println(new String(dis.readUTF()));
                        dis.close();
                        dos.close();
                        s.close();
                }
                catch(IOException e)
                {
                        System.out.println("IO: "+e.getMessage());
                }
        }
}

#Server

import java.util.*;
import java.net.*;
```

```
import java.io.*;

public class lab9server
{
        public static void main (String args[])
        {
                try
                {
                        ServerSocket s = new ServerSocket(998);
                        System.out.println("server ready\nwaiting for
connection\n");
                        Socket s1 = s.accept();
                        DataOutputStream dos = new
DataOutputStream(s1.getOutputStream());
                        DataInputStream dis = new
DataInputStream(s1.getInputStream());
                        System.out.println(dis.readUTF());
                        dos.writeUTF("connected to server\n");
                        String path = dis.readUTF();
                        System.out.println("\nrequest
received\nprocessing.......");
                        try
                        {
                                File myfile = new File(path);
                                Scanner scan = new Scanner(myfile);
                                String st = "\nthe context of file is\n" +
scan.nextLine();
                                while(scan.hasNextLine())
                                        st += "\n" + scan.nextLine();
                                dos.writeUTF(st);
                                dos.close();
                                s1.close();
                        }
                        catch(FileNotFoundException e)
                        {
                                System.out.println("\n...error...\nfile not
found");
                                dos.writeUTF("...error\nfile not found");
                        }
                }
                catch(IOException e)
                {
                        System.out.println("IO: " + e.getMessage());
                }
                finally
                {
                        System.out.println("\nconnection terminated");
                }
        }
}
```

#Program 10

#Client

```
import java.net.*;
import java.io.*;

public class lab10client
{
        public static void main(String args[])
        {
                DatagramSocket aSocket = null;
                try
                {
                        socket = new DatagramSocket(998);
                        byte[] buffer = new byte[1000];
                        DatagramPacket data = new DatagramPacket(buf,
buf.length);
                        System.out.println("Waiting for server\n");
                        socket.receive(data);
                        buffer = data.getData();
                        System.out.println("\n msg:" + (new String(buffer, 0,
data.getLength())));
                }
                catch(SocketException e)
                {
                        System.out.println("Socket:" + e.getMessage());
                }
                catch(IOException e)
                {
                        System.out.println("IO:" + e.getMessage());
                }
                finally
                {
                        if(socket != null)
                                socket.close();
                }
        }
}
```

#Server

```
import java.net.*;
import java.util.*;
import java.io.*;

public class lab10server
{
        public static void main(String args[])
        {
                DatagramSocket aSocket = null;
                Scanner scan = new Scanner(System.in);
                System.out.println("Server Ready\nWaiting for
connection....\n");
                try
                {
```

```
                        socket = new DatagramSocket(999);
                        byte[] buffer = new byte[1000];
                        System.out.println("\nEnter message to be sent:");
                        String str = scan.nextLine();
                        buffer = str.getBytes();
                        DatagramPacket data = new DatagramPacket(buffer,
buffer.length, InetAddress.getLocalHost(), 998);
                        socket.send(data);
                }
                catch(SocketException e)
                {
                        System.out.println("Socket:" + e.getMessage());
                }
                catch(IOException e)
                {
                        System.out.println("Io:" + e.getMessage());
                }
                finally
                {
                        System.out.println("\nMessage sent\nConnection
terminated");
                        if(socket != null)
                                socket.close();
                }
        }
}


#Program 11

import java.util.*;
import java.math.*;

public class lab11
{
        static int bitLength = 256;
        static BigInteger n;

        public static void main (String args[])
        {
                Scanner S = new Scanner(System.in);
                Random R = new Random();
                BigInteger p, q, e, d, phi;

                p = BigInteger.probablePrime(bitLength, R);
                q = BigInteger.probablePrime(bitLength, R);
                n = p.multiply(q);
                e = BigInteger.probablePrime(bitLength/2, R);
                phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

                while(phi.gcd(e).compareTo(BigInteger.ONE) != 0 &&
e.compareTo(phi) < 0)
                        e.add(BigInteger.ONE);
```

```
                d=e.modInverse(phi);

                System.out.print("Enter The Msg : ");
                byte msg[] = S.nextLine().getBytes();
                System.out.println("Msg Byte Array : " + display(msg));
                byte en[] = encryptdecrypt(msg, e);
                System.out.println("Encrypted Byte Array : " + display(en));
                byte de[] = encryptdecrypt(en, d);
                System.out.println("Decrypted Byte Array : " + display(de));
                System.out.println("Received Msg : " + new String(de));
        }

        static byte[] encryptdecrypt(byte a[], BigInteger key)
        {
                return (new BigInteger(a).modPow(key,n)).toByteArray();
        }

        static String display(byte a[])
        {
                String s = "";
                for(int i = 0; i < a.length; i++)
                        s += Byte.toString(a[i]);
                return s;
        }
}


#Program 12

import java.util.*;
public class lab12 {

        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                int i, bucket = 0;

                System.out.println("Enter the number of packets");
                int n = scan.nextInt();
                System.out.println("Enter the output rate of the bucket");
                int output = scan.nextInt();
                System.out.println("Enter the bucket size");
                int size = scan.nextInt();
                System.out.println("Enter the arriving packets(size)");
                int pkt[] = new int[n];
                for(i = 0; i < n; i++)
                        pkt[i] = scan.nextInt();

                System.out.println("\nSec\tPacket
Size\tBucket\tAccept/Reject\tPacket Sent");

System.out.println("-----------------------------------------------------");
                for(i = 0; i < n; i++)
                {
```

```
                        System.out.print(i+1 + "\t" + pkt[i] + "\t\t");
                        int x = bucket + pkt[i];
                        if(x <= size)
                        {
                                bucket = x;
                                System.out.print(bucket + "\tAccept\t\t" +
Math.min(bucket, output) + "\n" +"");
                        }
                        else
                        {
                                bucket = size;
                                System.out.print(bucket+"\tReject
"+(x-size)+"\t" + Math.min(bucket, output) + "\n");
                        }
                        bucket=drop(bucket,output);
                }
                while(bucket != 0)
                {
                        System.out.print(++i + "\t0\t\t" + bucket +
"\tAccept\t\t" + Math.min(bucket,output)+"\t");
                        bucket=drop(bucket,output);
                }
                System.out.println();
        }

        static int drop(int a,int b)
        {
                return a>b?(a-b):0;
        }
}
```