COMPUTER NETWORK LABORATORY

Subject Code: 15CSL57 IA Marks: 20
Number of Lecture Hours/Week: 01I + 02P Exam Marks: 80
Total Number of Lecture Hours: 40 Exam Hours: 03

Course objectives:

This course will enable students to

- Demonstrate operation of network and its management commands
- Simulate and demonstrate the performance of GSM and CDMA
- Implement data link layer and transport layer protocols.

Description (If any):

For the experiments below modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude. Use NS2/NS3.

Implement the programs in Java:

Course outcomes: The students should be able to:

- Analyze and Compare various networking protocols.
- Demonstrate the working of different concepts of networking.
- Implement, analyze and evaluate networking protocols in NS2 / NS3

Conduction of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from part A and part B with lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva: 80

   Part A: 10+25+5 =40

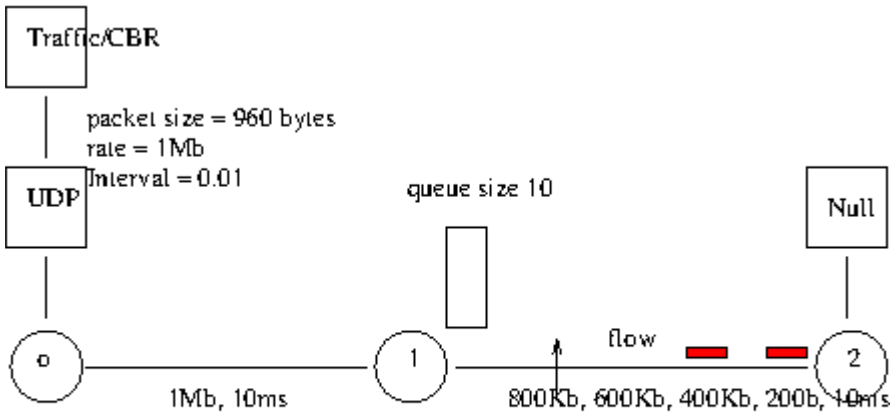
   Part B: 10+25+5 =40

5. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

CONTENTS

PART A
Simulation Exercises

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.**



Simulation time = 10 seconds

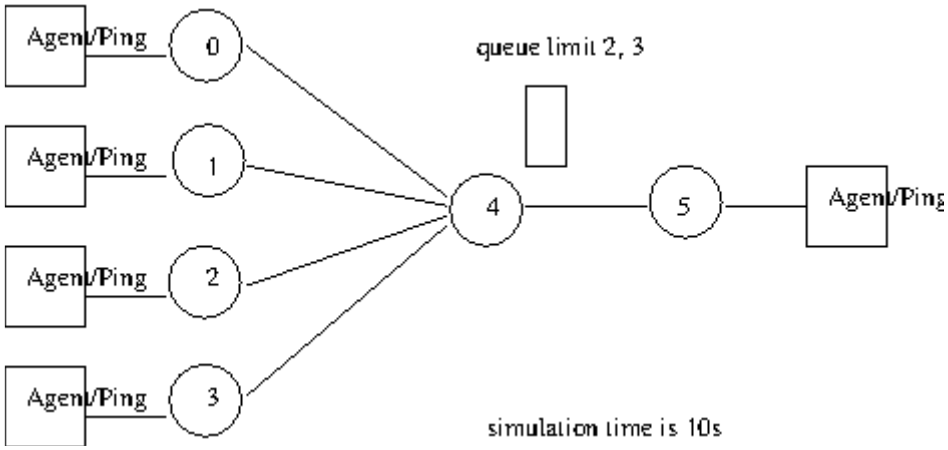
```
#
# File 1.tcl
# Three nodes network & measure packets dropped

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]

$ns trace-all $tf
$ns namtrace-all $nf

# Create nodes
set num 3
for {set i 0} {$i < $num} {incr i} {
    set node($i) [$ns node]
}

# Create links
$ns duplex-link $node(0) $node(1) 1Mb 10ms DropTail
$ns duplex-link $node(1) $node(2) 800Kb 10ms DropTail ;#800, 600, 400,
200

# Create queues
$ns duplex-link-op $node(1) $node(2) queuePos 0.5
$ns queue-limit $node(1) $node(2) 10

# Label nodes
$node(0) label "UDP"
$node(2) label "Null"

# Label flows
$ns color 0 Red
```

```
# Create connections
set udp [$ns create-connection UDP $node(0) Null $node(2) 0]
set cbr [$udp attach-app Traffic/CBR]

# Traffic
$cbr set packetSize_ 960
$cbr set rate_ 1Mb
$cbr set interval_ 0.001 ;# choose 0.01 only; 0.001, 0.01, 0.1

$ns at 0.0 "$cbr start"
$ns at 10 "finish"

proc finish {} {
      global ns tf nf
      $ns flush-trace
      close $tf
    close $nf
      exit 0
}

# Start simulation
$ns run
```

```
#
# File 1.awk
# Count dropped packets

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

```
RUN:
ns 1.tcl
nam out.nam
awk -f 1.awk out.tr
BW(Kb/s) 800 600 400 200
Dropped 0 210 470 730
```

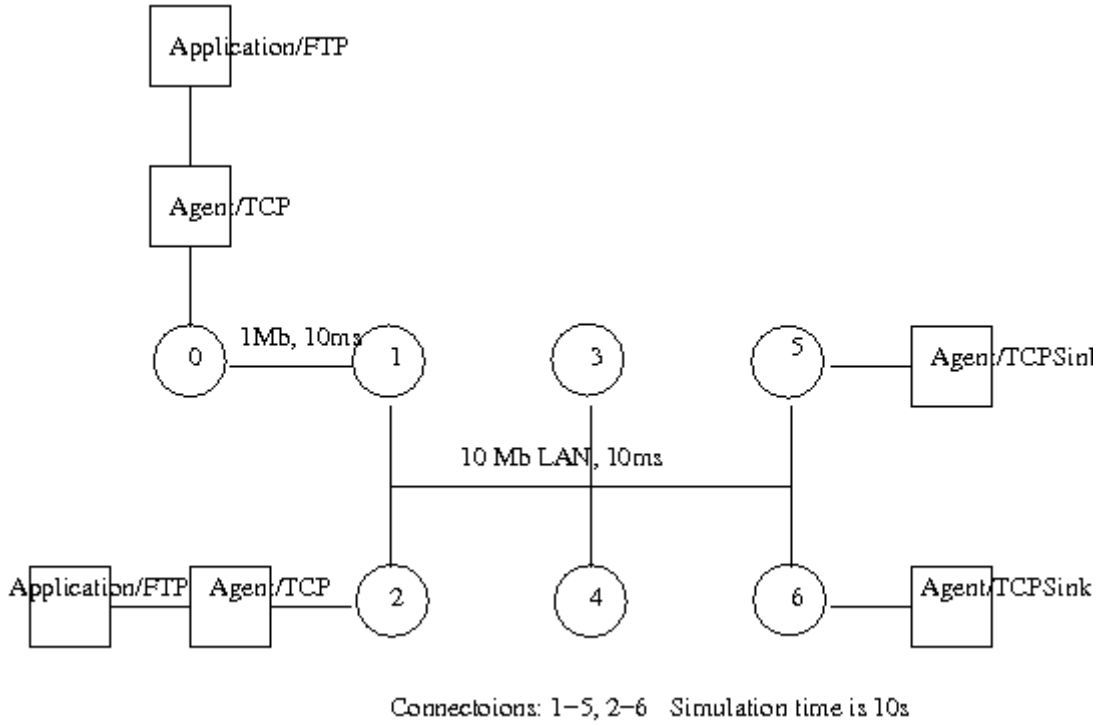
2. **Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.**




```
#
# File 2.tcl
# Simulate Ping & count dropped packets due to congestion

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]

$ns trace-all $tf
$ns namtrace-all $nf

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

# Create links
$ns duplex-link $node(0) $node(4) 1Mb 10ms DropTail
$ns duplex-link $node(1) $node(4) 1Mb 10ms DropTail
$ns duplex-link $node(2) $node(4) 1Mb 10ms DropTail
$ns duplex-link $node(3) $node(4) 1Mb 10ms DropTail
$ns duplex-link $node(4) $node(5) 1Mb 10ms DropTail

# Create queue
$ns duplex-link-op $node(4) $node(5) queuePos 0.5
$ns queue-limit $node(4) $node(5) 2 ;# different from normal 3, 2

# Label flows
$ns color 1 "red"
$ns color 2 "blue"
$ns color 3 "green"
$ns color 4 "yellow"
$ns color 5 "orange"

# Define a '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 connections
set p0 [$ns create-connection Ping $node(0) Ping $node(5) 1]
set p1 [$ns create-connection Ping $node(1) Ping $node(5) 2]
set p2 [$ns create-connection Ping $node(2) Ping $node(5) 3]
set p3 [$ns create-connection Ping $node(3) Ping $node(5) 4]
set p5 [$ns create-connection Ping $node(5) Ping $node(4) 5]

# Schedule events
for { set i 0} {$i < 10} {incr i} {
    for {set j 0} {$j < 10} {incr j} {
        $ns at [expr $i+.1+$j/10] "$p0 send"
        $ns at [expr $i+.1+$j/10] "$p5 send"
        $ns at [expr $i+.2+$j/10] "$p1 send"
        $ns at [expr $i+.3+$j/10] "$p2 send"
        $ns at [expr $i+.4+$j/10] "$p3 send"
        $ns at [expr $i+.5+$j/10] "$p5 send"
    }
}
$ns at 10 "finish"

proc finish {} {
      global ns tf nf
      $ns flush-trace
      close $tf
    close $nf
      exit 0
}

# Start simulation
$ns run

#
# File 2.awk
# Count dropped packets due to congestion

BEGIN {
    count=0;
}

{
    if($1=="d") count++;
}
END {
    printf("total no of packets dropped due to cngestion : %d\n",
count);
}

RUN:
ns 2.tcl
nam out.nam
awk -f 2.awk out.tr
1. qsize(n4,n5) = 2, 30 packets dropped due to congestion
2. qsize(n4,n5) = 3, 20 packets dropped
```

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

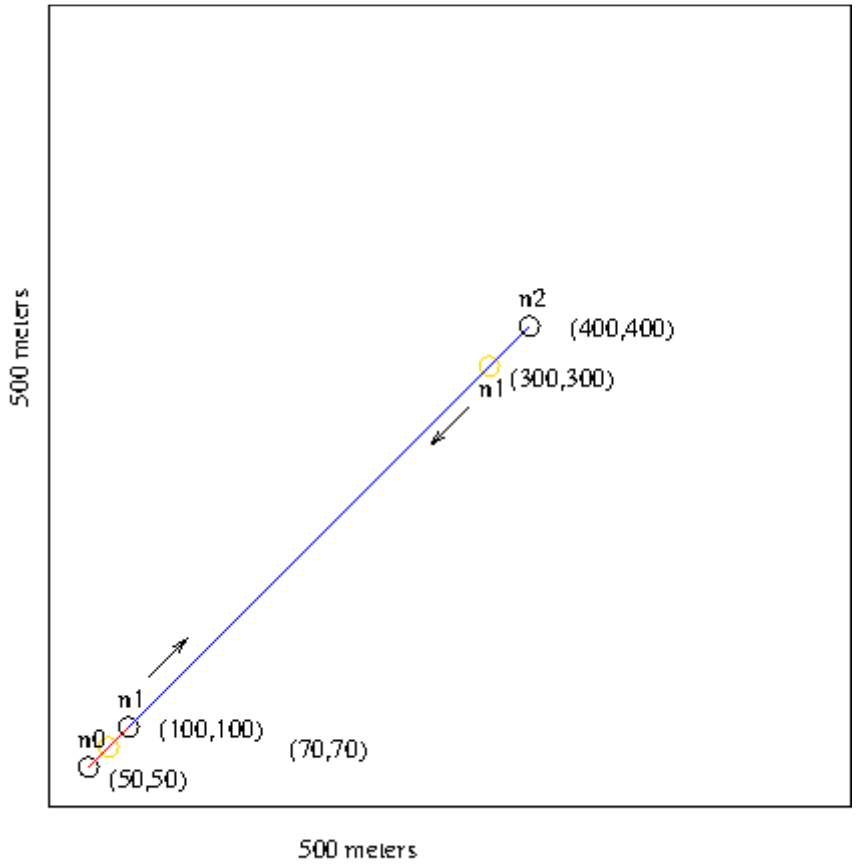


Connections: 1–5, 2–6 Simulation time is 10s

```
#
# File 3.tcl
# LAN simulation (congestion window size with time)

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]

$ns trace-all $tf
$ns namtrace-all $nf

# Create nodes
set node(0) [$ns node]

set num 6
for {set i 1} {$i <= $num} {incr i} {
    set node($i) [$ns node]
    lappend nodelist $node($i)
}

# create LAN and links
$ns make-lan $nodelist 10Mb 10ms LL Queue/DropTail

$ns duplex-link $node(0) $node(1) 1Mb 10ms DropTail
$ns duplex-link-op $node(0) $node(1) queuePos 0.5
$ns duplex-link-op $node(0) $node(1) orient right

# Create connections
set tcp0 [$ns create-connection TCP $node(0) TCPSink $node(5) 0]
set tcp1 [$ns create-connection TCP $node(2) TCPSink $node(6) 0]
```

```
set ftp0 [$tcp0 attach-app FTP]
set ftp1 [$tcp1 attach-app FTP]

$tcp0 attach $tf
$tcp0 trace cwnd_

$tcp1 attach $tf
$tcp1 trace cwnd_

$ns at 0.1 "$ftp0 start"
$ns at 0.2 "$ftp1 start"

$ns at 10 "finish"

proc finish {} {
      global ns tf nf
      $ns flush-trace
      close $tf
    close $nf
      exit 0
}

# Start simulator
$ns run

#
# File 3.awk
# Plot congestion window X time

BEGIN{
}
{
    if($6=="cwnd_")
    {
        if ($2 == 0 && $4 == 5) printf("%4.2f\t%4.2f\t\n",$1,$7);
# $1=time, $7=cwnd size
#       if ($2 == 2 && $4 == 6) printf("%4.2f\t%4.2f\t\n",$1,$7);
    }
}
END{
    puts "DONE";
}

RUN:
ns 3.tcl
nam out.nam
awk -f 3.awk out.tr > out.txt
xgraph out.txt
modify awk script to use another tcp connection
```

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



Simulation time 25 s; n1 moves towards n2 at 10s; retracts back at 20 s.

```
#
# File 4.tcl
# Wireless LAN simulation

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]

$ns trace-all $tf
$ns namtrace-all-wireless $nf 500 500

set topo [new Topography]
$topo load_flatgrid 500 500

$ns node-config \
    -adhocRouting DSDV \
    -llType       LL \
    -macType      Mac/802_11 \
    -ifqType      Queue/DropTail \
    -ifqLen       10 \
    -phyType      Phy/WirelessPhy \
    -propType     Propagation/TwoRayGround \
    -antType      Antenna/OmniAntenna \
    -topoInstance $topo \
    -agentTrace   ON \
    -routerTrace  ON \
```

```
    -macTrace     ON \
    -channel      [new Channel/WirelessChannel]

create-god 3 ;# General Operations Director

set num 3
for {set i 0} {$i < $num} {incr i} {
    set node($i) [$ns node]
}

$node(0) label "TCP"
$node(1) label "TCPSink, TCP"
$node(2) label "TCPSink"

$node(0) set X_ 50
$node(0) set Y_ 50
$node(0) set Z_ 0

$node(1) set X_ 100
$node(1) set Y_ 100
$node(1) set Z_ 0

$node(2) set X_ 400
$node(2) set Y_ 400
$node(2) set Z_ 0

# Create connections
set tcp0 [$ns create-connection TCP $node(0) TCPSink $node(1) 1]
set tcp1 [$ns create-connection TCP $node(1) TCPSink $node(2) 2]

$ns color 1 "red"
$ns color 2 "blue"

set ftp0 [$tcp0 attach-app FTP]
set ftp1 [$tcp1 attach-app FTP]

$ns at 0 "$node(0) setdest 50 50 100"
$ns at 0 "$node(1) setdest 100 100 100"
$ns at 0 "$node(2) setdest 400 400 100"

$ns at 1 "$ftp0 start"
$ns at 1 "$ftp1 start"

$ns at 10 "$node(1) setdest 300 300 100"
$ns at 15 "$node(1) setdest 100 100 100"

$ns at 20 "finish"

proc finish {} {
      global ns tf nf
      $ns flush-trace
      close $tf
    close $nf
      exit 0
}

# Start simulation
$ns run
```

```
#
# File 4.awk
# Wireless LAN link performance

BEGIN{
      count1=0;
      count2=0;
      pack1=0;
      pack2=0;
      time1=0;
      time2=0;
}
{
      if($1=="r" && $3=="_1_" && $4=="AGT")
      {
            count1++;
            pack1=pack1+$8
            time1=$2;
      }
      if($1=="r" && $3=="_2_" && $4=="AGT")
      {
            count2++;
            pack2=pack2+$8;
            time2=$2;
      }
}
END{
      printf("node(0) to node(1) link performance : %6.2f
Mbps\n",((count1*pack1*8)/(time1*1000000)));
      printf("node(0) to node(1) link performance : %6.2f
Mbps\n",((count2*pack2*8)/(time2*1000000)));
}

RUN:
ns 4.tcl
nam out.nam
awk -f 4.awk out.tr
The throughput from node(0) to node(1): 415.40 Mb/s
The throughput from node(1) to node(2): 184.56 Mb/s
```

5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.

```
#
# File 5.tcl
# Wireless LAN simulation

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf
set node(ms1) [$ns node]
set node(bs1) [$ns node]
set node(msc) [$ns node]
set node(bs2) [$ns node]
set node(ms2) [$ns node]

$ns duplex-link $node(ms1) $node(bs1) 1Mb 1ms DropTail
$ns duplex-link $node(bs1) $node(msc) 1Mb 10ms DropTail
$ns duplex-link $node(msc) $node(bs2) 1Mb 10ms DropTail
$ns duplex-link $node(bs2) $node(ms2) 1Mb 1ms DropTail

puts "Cell Topology"

$ns bandwidth $node(ms1) $node(bs1) 9.6Kb simplex
$ns bandwidth $node(bs1) $node(ms1) 9.6Kb simplex

$ns insert-delayer $node(ms1) $node(bs1) [new Delayer]


set tcp1 [$ns create-connection TCP $node(ms1) TCPSink $node(ms2) 0]
set ftp1 [$tcp1 attach-app FTP]
$ns at 0.1 "$ftp1 start"


proc stop {} {
      global node opt nf
      set sid [$node(ms1) id]
      set did [$node(bs1) id]

    puts $sid
    puts $did

    exec getrc -s $sid -d $did -f 0 out.tr | \
         raw2xg -s 0.01  -m 100 -r > plot.xgr
    exec getrc -s $did -d $sid -f 0 out.tr | \
         raw2xg -a -s 0.01 -m 100  >> plot.xgr

      exec ./xg2gp.awk plot.xgr
    exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
      exit 0
}
$ns at 20 "stop"
$ns run
```

6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

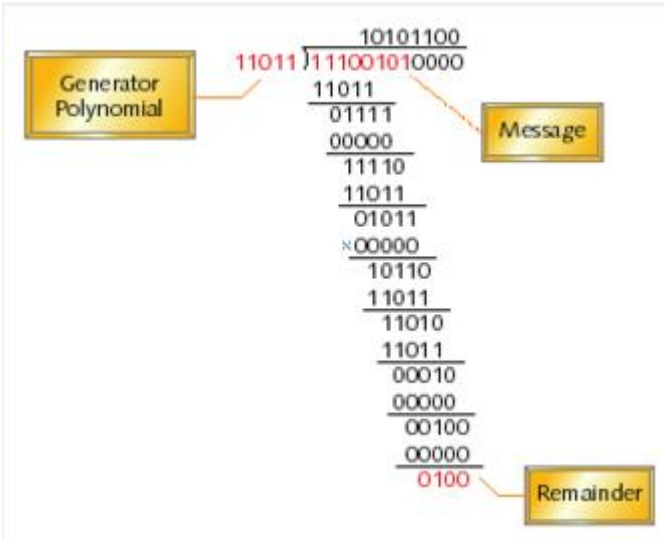
```
#
# File 6.tcl
# Wireless LAN simulation

set ns [new Simulator]
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf
set node(ms1) [$ns node]
set node(bs1) [$ns node]
set node(msc) [$ns node]
set node(bs2) [$ns node]
set node(ms2) [$ns node]

$ns duplex-link $node(ms1) $node(bs1) 1Mb 1ms DropTail
$ns duplex-link $node(bs1) $node(msc) 1Mb 10ms DropTail
$ns duplex-link $node(msc) $node(bs2) 1Mb 10ms DropTail
$ns duplex-link $node(bs2) $node(ms2) 1Mb 1ms DropTail

puts "Cell Topology"

$ns bandwidth $node(ms1) $node(bs1) 64Kb simplex
$ns bandwidth $node(bs1) $node(ms1) 384Kb simplex

$ns insert-delayer $node(ms1) $node(bs1) [new Delayer]


set tcp1 [$ns create-connection TCP $node(ms1) TCPSink $node(ms2) 0]
set ftp1 [$tcp1 attach-app FTP]
$ns at 0.1 "$ftp1 start"


proc stop {} {
      global node opt nf
      set sid [$node(ms1) id]
      set did [$node(bs1) id]

    puts $sid
    puts $did

    exec getrc -s $sid -d $did -f 0 out.tr | \
         raw2xg -s 0.01  -m 100 -r > plot.xgr
    exec getrc -s $did -d $sid -f 0 out.tr | \
         raw2xg -a -s 0.01 -m 100  >> plot.xgr

      exec ./xg2gp.awk plot.xgr
    exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
      exit 0
}
$ns at 20 "stop"
$ns run
```

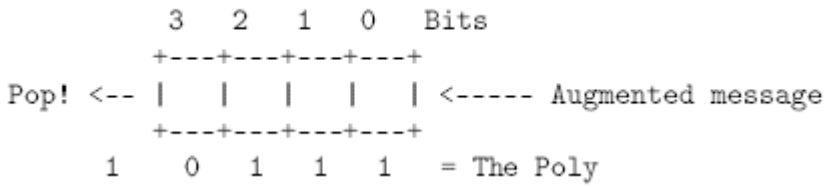
Part B

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

Theory:

It does error checking via polynomial division.





Load the register with zero bits.
Augment the message by appending W zero bits to the end of it.
While (more message bits)
Begin
Shift the register left by one bit, reading the next bit of the augmented message into register bit position 0.
If (a 1 bit popped out of the register during shifting)
Register = Register XOR Poly.
End
The register now contains the remainder.

```
import java.util.Scanner;

public class CRC {
      int W;
      char[] P;
      String checksum;
      String message;

      CRC()
```

```
        {
      W = 16;
P = "10001000000100001".toCharArray(); // (16,12,5,0) ["CRC-16"]
        }

      void crc()
      {
            String crc = "0000000000000000"; // W zeros
            char[] msg = (message + crc).toCharArray(); // augmented
            //message
            char[] rem = (crc+'0').toCharArray();

            int n = 0;
            while (n < msg.length)
            {
                  rem[W] = msg[n++];
                  boolean xorcopy = rem[0] == '1';
               for (int i=1; i <= W; i++)
                 {
               rem[i-1] = xorcopy ? (rem[i]==P[i]?'0':'1') : rem[i];
                 }

            }
            checksum = String.valueOf(rem).substring(0, W);
      }

      void input()
      {
            Scanner scanner = new Scanner(System.in);

            System.out.print("MESSAGE:");
            message = scanner.next();

            scanner.close();
       }

      void output()
      {
            System.out.println("Checksum:"+checksum);
      }


      public static void main(String[] args)
      {
            CRC crc = new CRC();

            crc.input();
            crc.crc();
            crc.output();
      }
}
```

Output

```
RUN:
javac CRC.java
java CRC
MESSAGE: 0101
Checksum: 0101000010100101

MESSAGE: 1011101
Checksum: 1000101101011000

MESSAGE: MESSAGE + Checksum.
Checksum: 0000000000000000

MESSAGE: (MESSAGE + CHecksum) error bits
Checksum: NOT Zero
```

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

DESIGN

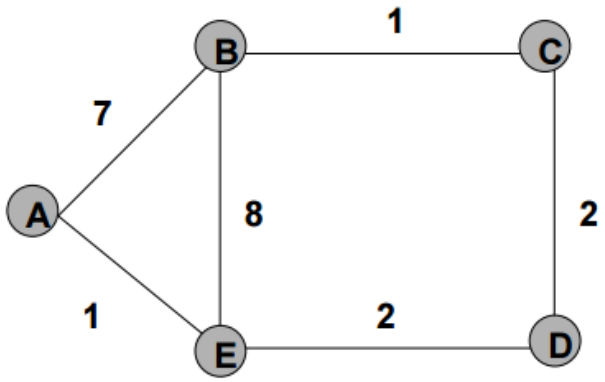
```
BELLMAN-FORD(G, w, s)
// Initialization
for each vertex v 2 G.V
v.d = 1
v._ = NIL
s.d = 0
// Relaxation
for i = 1 to |G.V|-1
for each edge(u,v) 2 G.E
if v.d > u.d + w(u,v)
v.d = u.d + w(u,v)
v._ = u
for each edge(u,v) 2 G.E
if v.d > u.d + w(u,v)
return FALSE
return TRUE;
```

CODE:

```
import java.util.Scanner;

public class BellmanFord {
      int n;
      int[][] a;
      int[]      d;
      int[]      p;
      int s;
      public final static int INFTY=999;

      BellmanFord(int n)
      {
            this.n = n;

            a = new int[n][n];
            d = new int[n];
            p = new int[n];
      }

      void bellmanFord()
      {
         // Initialization
            for(int i=0; i < n; i++)
            {
                  d[i] = a[s][i];
                  p[i] = a[s][i] == INFTY ? -1 : s;
            }
            p[s] = -1;

            for(int i=0; i < n-1; i++)
            {
             // Relax all edges iteratively (n-1) times
                  for(int u=0; u < n; u++)
```

```
                {
                        for(int v=0; v < n; v++)
                        {
                                if(d[v] > d[u]+a[u][v])
                                {
                                        d[v] = d[u]+a[u][v];
                                        p[v] = u;
                                }
                        }
                }
        }
    }

    void input(Scanner scanner)
    {
        System.out.println("Enter G: ");

        for(int i=0; i<n; i++)
        {
            for(int j=0; j<n; j++)
            {
                a[i][j] = scanner.nextInt();
                if (i != j && a[i][j] == 0) a[i][j] = INFTY;
            }
        }

        System.out.print("Enter the source vertex: ");
        s = scanner.nextInt();

        scanner.close();
    }

    void path(int v)
    {
        if (v == -1) return;

        path(p[v]);
        System.out.print("."+v);
    }

    void output()
    {
        int i;

        for(i=0; i < n; i++)
        {
            System.out.print("d(" + s + "," + i + ")=" +
d[i]+" :p");
            path(i);
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int        n;
```

```
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter n: ");
            n = scanner.nextInt();

            BellmanFord bf = new BellmanFord(n);

            bf.input(scanner);
            bf.bellmanFord();
            bf.output();
      }
}
```




```
javac BellmanFord.java
java BellmanFord
INPUT:
Enter n: 5
Enter a:
0 7 0 0 1
7 0 1 0 8
0 1 0 2 0
0 0 2 0 2
1 8 0 2 0
Dest Dist path...
1 6 4.3.2.1.
2 5 4.3.2.
3 3 4.3.
4 1 4.
```

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.

DESIGN:
SERVER
. Create a server socket and bind to a speci_c address
. Wait for client connection
. Create input and output stream for the client socket
. Read _le name from the input stream
. Read all lines from the _le
. Write the lines to output stream
CLIENT
. Create a socket with the server address
. Create input and output stream for the socket
. Read _le name from the console
. Write _le name to the socket
. Read, in a loop, the lines from server until the line is "stop"

CODE: TcpServer.java, TcpClient.java, f.txt

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

import java.nio.file.*;
import java.nio.charset.*;
import java.util.*;

public class TcpServer {

      void server() throws Exception
      {
            ServerSocket ss=new ServerSocket(3333);
         System.out.println("Server waiting for connection from
client");

            Socket cs=ss.accept();

      DataInputStream din=new DataInputStream(cs.getInputStream());
DataOutputStream dout=new DataOutputStream(cs.getOutputStream());

            String fileName = din.readUTF();
            List<String> lines =
Files.readAllLines(Paths.get(fileName), Charset.defaultCharset());

            for (int i=0; i < lines.size(); i++)
            {
                  System.out.println("server: "+lines.get(i));
                  dout.writeUTF(lines.get(i));
            }

            din.close();
```

```
            cs.close();
            ss.close();
      }

      public static void main(String[] args) throws Exception {
            TcpServer ts = new TcpServer();
            ts.server();
      }

}

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

public class TcpClient {
      void client() throws Exception {
            Socket s=new Socket("localhost",3333);

DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter filename:");
            String fileName = scanner.next();

            dout.writeUTF(fileName);
            String message;
            do
            {
                  message = din.readUTF();
                  System.out.println("Client: " + message);
            }
            while(!message.equals("stop"));

            scanner.close();

            dout.close();
            s.close();
      }

      public static void main(String[] args) throws Exception {
            TcpClient tc = new TcpClient();
            tc.client();
      }
}

RUN:
javac TcpServer.java TcpClient.java
java TcpServer
java TcpClient
Enter Filename: f.txt

Hello this is the sample text from the server
Displayed at client end
```

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

DESIGN:

SERVER
. Create a datagram socket and bind to a specific address
. Create a datagram packet for given message buffer
. Receive datagram packet and extract the message and client address
. Read line from the console and write to the socket until "stop"

CLIENT
. Create a datagram socket
. Create a datagram packet with message, server address
. Send the packet
. Receive, in a loop, packet and display the message until the "stop"

Program

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

class UdpServer
{
   public void server() throws Exception
   {
      DatagramSocket socket = new DatagramSocket(3333);
      DatagramPacket packet;

      System.out.println("UDP Server Listening in " + 3333);

      byte[] msgBuffer = new byte[1024];
      packet = new DatagramPacket(msgBuffer, msgBuffer.length);
      socket.receive(packet);

      String message = new String(msgBuffer, 0, packet.getLength());
      System.out.println("Client: Message received = " + message);

      InetAddress address = packet.getAddress();
      int port = packet.getPort();

      Scanner scanner = new Scanner(System.in);
      System.out.println("Server: type lines of text; type 'stop' to terminate");

      do
         {
            message = scanner.nextLine();
            packet = new DatagramPacket(message.getBytes(), message.length(), address,
port);
```

```
            socket.send(packet);
          }
       while (message.compareTo("stop") != 0);
       scanner.close();
       socket.close();
   }

   public static void main(String args[]) throws Exception
   {
      UdpServer us = new UdpServer();
      us.server();
   }
}

import java.net.*;
class UdpClient
{
   public void client() throws Exception
   {
      DatagramSocket socket = new DatagramSocket();
      DatagramPacket packet;
     String message = "Start";
      packet = new DatagramPacket(message.getBytes(), message.length(),
InetAddress.getByName("localhost"), 3333);

      socket.send(packet);
      System.out.println("Client: Sent data to Server ; Message = " + message);
      byte[] msgBuffer = new byte[1024];
      packet = new DatagramPacket(msgBuffer, msgBuffer.length);

      do
         {
            socket.receive(packet);

            message = new String(msgBuffer, 0, packet.getLength());
            System.out.println(message);
         }
      while (!message.equals("stop"));
      socket.close();
   }

   public static void main(String args[]) throws Exception
   {
      UdpClient uc = new UdpClient();
      uc.client();
   }
}
```

RUN

javac UdpServer.java UdpClient.java

java UdpServer
java UdpClient

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

DESIGN

Algorithm

1. Generate two large random primes, P and Q, of approximately equal size.
2. Compute $N = P \times Q$
3. Compute $Z = (P-1) \times (Q-1)$.
4. Choose an integer E, $1 < E < Z$, such that GCD $(E, Z) = 1$
5. Compute the secret exponent D, $1 < D < Z$, such that $E \times D \equiv 1 \pmod{Z}$
6. The public key is (N, E) and the private key is (N, D).

Note: The values of P, Q, and Z should also be kept secret.
The message is encrypted using public key and decrypted using private key.
Encryption: $C = P(M) = M^e \pmod{n}$; Decryption: $M = S(C) = C^d \pmod{n}$

An example of RSA encryption

1. Select primes P=11, Q=3.
2. $N = P \times Q = 11 \times 3 = 33$
   $Z = (P-1) \times (Q-1) = 10 \times 2 = 20$
3. Lets choose E=3
   Check GCD(E, P-1) = GCD(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1), and check GCD(E, Q-1) = GCD(3, 2) = 1
   therefore GCD(E, Z) = GCD(3, 20) = 1
4. Compute D such that $E \times D \equiv 1 \pmod{Z}$
   compute $D = E^{\wedge -1} \bmod Z = 3^{\wedge -1} \bmod 20$
   find a value for D such that Z divides (($E \times D$)-1)
   find D such that 20 divides 3D-1.
   Simple testing (D = 1, 2, ...) gives D = 7
   Check: ($E \times D$)-1 = 3.7 - 1 = 20, which is divisible by Z.
5. Public key = (N, E) = (33, 3)
   Private key = (N, D) = (33, 7).

Now say we want to encrypt the message m = 7,
Cipher code = $M^{\wedge E} \bmod N$
$= 7^{\wedge 3} \bmod 33$
$= 343 \bmod 33$
$= 13.$
Hence the ciphertext c = 13.

To check decryption we compute Message' = $C^{\wedge D} \bmod N$
$= 13^{\wedge 7} \bmod 33$
$= 7.$

CODE

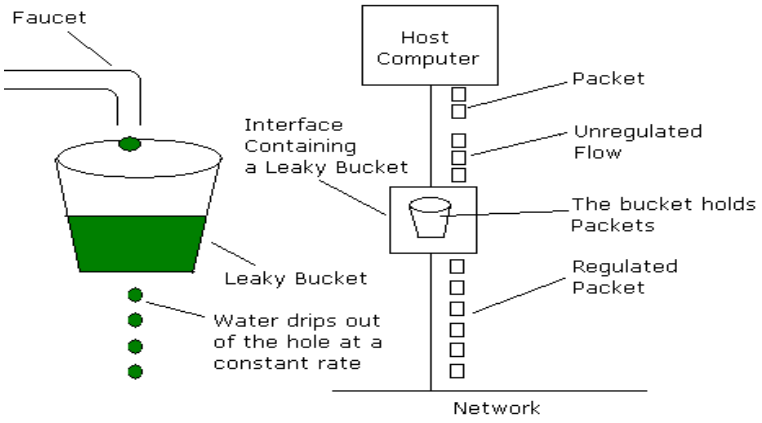
```
import java.util.Scanner;

public class RSA {
	int d;
	int e;
	int n;

	String  M;

	int gcd(int m, int n)
	{
		int rv = n == 0 ? m : gcd(n, m % n);

		return rv;
	}

	int pow(int a, int m, int n)
	{
		int r = 1;
		while (m-- != 0)
			r = (r*a) % n;

		return r ;
	}

	void rsa()
	{
		int p;
		int q;
		int z;

		//odd prime numbers 3, 5 (3, 11?)
		p = 11;
		q = 13;

		n = p*q;
		z = (p-1)*(q-1);

		// choose e
		for (e=2; gcd(e, z) != 1; e++);

		// choose d as inverse of e
		for (d=2; (d*e) % z != 1; d++);
	}

	void input()
	{
		Scanner scanner = new Scanner(System.in);
```

```
                System.out.print("Enter M:");
                M = scanner.next();

                scanner.close();
        }

        void output()
        {
                System.out.println("S=(d,n)=(" + d + "," + n + ")");
                System.out.println("P=(e,n)=(" + e + "," + n + ")");

                System.out.println("T: " + M);

    char[] T = M.toCharArray();
    for (int i=0; i < M.length(); i++)
        {
           T[i] = (char)pow((int)T[i], e, n);
        }
                System.out.println("C: " + String.valueOf(T));

    for (int i=0; i < M.length(); i++)
        {
           T[i] = (char)pow((int)T[i], d, n); //convert(a, d, n);
        }
                System.out.println("T: " + String.valueOf(T));
        }


        public static void main(String[] args) {
                RSA r = new RSA();

                r.input();
                r.rsa();
                r.output();
        }
}
```

```
RUN:
javac RSA.java
java RSA
Enter M:ABCabc123
S=(d,n)=(103,143)
P=(e,n)=(7,143)
T: ABCabc123
C: ABY; ,$)t
T: ABCabc123
```

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

DESIGN:



- The leaky bucket algorithm used to control rate in a network
- In this algorithm the input rate can vary but the output rate remains constant
- This algorithm saves bursty tra_c into _xed rate tra_c by averaaging the data rate
- If the bucket (bu_er) over_ows then packets are discarded
- It is implemented as a single-server queue with constant service rate

ALGORITHM

Step 1. Initialise the counter to 'n' at every tick of clock

Step 2. If n is greater than the size of packet in the front of queue send the packet into the network and decrement the counter by size of packet.

Repeat the step until 'n' is less than the size of packet.

Step 3. Reset the counter and goto step 1.

n = 1000, buffer size of 6 packets.
incoming packets _> [200,700,500,450,400,200] -> sent packets
packets 200, 400 are sent and n=400. Next packet will not be sent as its size is > n. Wait for the next clock tick before reseting n.

CODE

```
import java.util.*;

public class LeakyBucket {

        int n;
        int burst;
        int outgoingRate;
        int bucketSize;
```

```
int incoming;
int outgoing;
int pending;
int overflow;

int duration;
int interval;

LeakyBucket()
{
        pending = 0;
        incoming = 0;
        overflow = 0;
        outgoing = 0;
}


void leakyBucket()
{
        System.out.println("Time Incoming Pending Overflow Outgoing");

        Random rand = new Random();
        int time=0;
        while (time < duration)
        {
                incoming = rand.nextInt(burst);
                if ((pending + incoming) > bucketSize)
                {
                        overflow = (pending + incoming) - bucketSize;
                        pending = bucketSize;
                }
                else pending += incoming;

                //interval = rand()%10; // Next packet will come at time
                interval = 1;

                for(int clk = 0; clk < interval; ++clk)
                {
                        output(time, incoming, pending, overflow, outgoing);
                        //sleep(1);
                        outgoing = Math.min(outgoingRate, pending);
                        pending -= outgoing;

                        incoming = 0;
                        ++time;
                }
        }
}

void input()
```

```
        {
                Scanner scanner = new Scanner(System.in);
                System.out.println("Enter burst size: ");
                burst = scanner.nextInt();

                System.out.println("Enter bucket size: ");
                bucketSize = scanner.nextInt();

                System.out.println("Enter outgoing rate: ");
                outgoingRate = scanner.nextInt();

                System.out.println("Enter duration: ");
                duration = scanner.nextInt();

                scanner.close();
        }

        void output(int time, int incoming, int pending, int overflow, int outgoing)
        {

        System.out.printf("%d\t%d\t%d\t%d\t%d\n",time,incoming,pending,overflow,outgo
ing);
        }


        public static void main(String[] args) {
                LeakyBucket lb = new LeakyBucket();

                lb.input();
                lb.leakyBucket();
        }

}
```

RUN

```
javac LeakyBucket.java
java LeakyBucket
Enter burst size: 8
Enter bucket size: 8
Enter outgoing rate: 2
Enter duration: 8
Time Incoming Pending Overflow Outgoing
0       2       2               0       0
1       4       4               0       2
2       5       7               0       2
3       4       8               1       2
4       5       8               3       2
5       6       8               4       2
6       5       8               3       2
7       0       6               3       2
```