COMPUTER NETWORK LABORATORY
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2016 -2017)

SEMESTER – V

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

CREDITS – 02

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.

Lab Experiments:

<u>PART A</u>

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. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

PART B

Implement the following in Java:

7. Write a program for error detecting code using CRC-CCITT (16- bits).
8. Write a program to find the shortest path between vertices using bellman-ford algorithm.
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.
10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
11. Write a program for simple RSA algorithm to encrypt and decrypt the data.
12. Write a program for congestion control using leaky bucket algorithm.

Study Experiment / Project:

NIL

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.

Q1. Simulate three nodes point-to-point networks with a duplex link between them. Set the queue size and vary the bandwidth and find the number of packets dropped.

```
#===================================
#        Initialization
```

```
#=====================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#=====================================
#        Nodes Definition
#=====================================
#Create 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
#=====================================
#        Links Definition
#=====================================
#Createlinks between nodes
$ns duplex-link $n0 $n1 10.0Mb 1ms DropTail
$ns queue-limit $n0 $n1 10
$ns duplex-link $n1 $n2 10.0Mb 1ms DropTail
$ns queue-limit $n1 $n2 10

#Give node position (for NAM)
#$ns duplex-link-op $n0 $n1 orient right
#$ns duplex-link-op $n1 $n2 orient right

#######################
#1.to create drop scenario at first node itself ----- >> change the
packet size of application protocol and packet size of Transport
# layer e.g packet size of cbr =10000 , packet size of tcp =100
#2. Drop at n1 = set queue size ratio to be 5:2 , BWXDelay between no
and n1 = 10Mb X 0.05ms , between n1 and n2 0.05Mb X 100ms
#3 . to count the number of packets dropped grep -c "^d" out.tr
```

```
#=====================================
#        Agents Definition
#=====================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns connect $tcp0 $sink1
```

```
$tcp0 set packetSize_ 1500

#===================================
#        Applications Definition
#===================================
#Setup a CBR Application over TCP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $tcp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"

#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}

$ns at 10.0 "finish"

$ns run
```

OUTPUT:-

```
student@unixlab:~$ gedit 1.tcl
student@unixlab:~$ ns 1.tcl

student@unixlab:~$ grep -c "^d" out.tr
0
```

```
student@unixlab:~$ grep -c “^d” out.tr
8
```

Q2. Simulate a transmission of ping message over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

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

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

$ns color 1 Red
$ns color 2 Green
#====================================
#        Nodes Definition
#====================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#====================================
#        Links Definition
#====================================
#Createlinks between nodes
$ns duplex-link $n0 $n1 10.0Mb 0.05ms DropTail
$ns queue-limit $n0 $n1 5
$ns duplex-link $n1 $n2 0.05Mb 100ms DropTail
$ns queue-limit $n1 $n2 2
$ns duplex-link $n2 $n3 10.0Mb 1ms DropTail
$ns queue-limit $n2 $n3 10
$ns duplex-link $n3 $n4 10.0Mb 1ms DropTail
$ns queue-limit $n3 $n4 10
$ns duplex-link $n4 $n5 10.0Mb 1ms DropTail
$ns queue-limit $n4 $n5 10

## to create congestion and to depict the packet drop
# 1. BW X Delay [n0->n1 10MB X 0.05 ms ,Queue Size =5 ] + [ n1->n2
0.05Mb X 100 ms , Queue size =2 ]
#               add 4 sends from p0 at 1.0 , similarly add 4 sends
from p2 at 1.0 === drop at n1
# repeat the same scenario for p2 , p3 , p4 and p5 to create
congestion scenario




#Give node position (for NAM)
#$ns duplex-link-op $n0 $n1 orient right
#$ns duplex-link-op $n1 $n2 orient right
```

```
#$ns duplex-link-op $n2 $n3 orient right-down
#$ns duplex-link-op $n3 $n4 orient left
#$ns duplex-link-op $n4 $n5 orient left

Agent/Ping instproc recv {from rtt} {
      $self instvar node_
      puts "node [$node_ id] received ping answer from \
              #$from with round-trip-time $rtt ms."
}

#===================================
#        Agents Definition
#===================================
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0

$p0 set fid_ 1

set p1 [new Agent/Ping]
$ns attach-agent $n5 $p1

$p1 set fid_ 2

#Connect the two agents
$ns connect $p0 $p1
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
# to create drop at n1 following sends
$ns at 0.2 "$p0 send"
$ns at 0.2 "$p0 send"
$ns at 0.2 "$p0 send"
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 2.0 "finish"
$ns run
```

<u>OUTPUT:-</u>

```
student@unixlab:~$ gedit 2.tcl
```

```
student@unixlab:~$ ns 2.tcl
node 0 received ping answer from  #5 with round-trip-time 227.0 ms.
node 0 received ping answer from  #5 with round-trip-time 237.2 ms.
node 5 received ping answer from  #0 with round-trip-time 227.0 ms.
node 5 received ping answer from  #0 with round-trip-time 237.2 ms.
node 5 received ping answer from  #0 with round-trip-time 247.5 ms.
node 5 received ping answer from  #0 with round-trip-time 257.7 ms.


student@unixlab:~$ grep -c “^d” out.tr
2
```

Q3. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source/destination

```
set ns [new Simulator]

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

## The code you need to add -Change 1
set winFile0 [open WinFile0 w]
set winFile1 [open WinFile1 w]

#===================================
#        Nodes Definition
#===================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#===================================
#        Links Definition
#===================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 10.0Mb 1ms DropTail
$ns queue-limit $n0 $n2 10
$ns duplex-link $n1 $n2 10.0Mb 1ms DropTail
$ns queue-limit $n1 $n2 10
$ns simplex-link $n2 $n3 10.0Mb 1ms DropTail
$ns queue-limit $n2 $n3 10
$ns simplex-link $n3 $n2 10.0Mb 1ms DropTail
$ns queue-limit $n3 $n2 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left

## change 2 -setting up the lan
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail
MAC/802_3 Channel]
#===================================
#        Agents Definition
#===================================

#Setup a TCP/Newreno connection
set tcp0 [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
```

```
$ns attach-agent $n4 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500
$tcp0 set window 5000   # change 3 -set the tcp window size

#Setup a TCP/Newreno connection
set tcp1 [new Agent/TCP/Newreno]
$ns attach-agent $n5 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n1 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500
$tcp1 set window 500   # change 4 -set the tcp window size

#===================================
#        Applications Definition
#===================================
#Setup a FTP Application over TCP/Newreno connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"

#Setup a FTP Application over TCP/Newreno connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"

# change 4 -setting up error model between $n2 $ n3 in random fashion
set var [new ErrorModel]
$var ranvar [new RandomVariable/Uniform]
$var drop-target [new Agent/Null]
$ns lossmodel $var $n2 $n3
#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
              exec xgraph WinFile0 WinFile1 &   # change 5 executing
x-graph
    exit 0
}

# change 6 adding plot window function
proc PlotWindow {tcpSource file} {
global ns
set time 0.1              # increment =0.1
set now [$ns now]  # it will set now -> current time
set cwnd [$tcpSource set cwnd_] # set the window of tcp to tcp1 & tcp2
puts $file "$now $cwnd"   # file contains 2 values time & Congestion
#Window
$ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}

# change 7 schedule it
$ns at 0.1 "PlotWindow $tcp0 $winFile0"
$ns at 0.1 "PlotWindow $tcp1 $winFile1"
$ns at 10.0 "finish"

$ns run
```

Q4.Simulate simple ESS with transmitting nodes in wireless LAN by simulation and determine the performance w.r.t transmission of packets

```
set opt(chan)   Channel/WirelessChannel        ;# channel type
set opt(prop)   Propagation/TwoRayGround       ;# radio-propagation
model
set opt(netif)  Phy/WirelessPhy               ;# network interface type
set opt(mac)    Mac/802_11                     ;# MAC type
set opt(ifq)    Queue/DropTail/PriQueue        ;# interface queue type
set opt(ll)     LL                             ;# link layer type
set opt(ant)    Antenna/OmniAntenna            ;# antenna model
set opt(ifqlen)         50                     ;# max packet in ifq
set opt(nn)             1                     ;# number of mobilenodes
set opt(adhocRouting)   DSDV                   ;# routing protocol
set opt(x)      670                        ;# x coordinate of topology
set opt(y)      670                        ;# y coordinate of topology
set opt(seed)   0.0                            ;# random seed
set opt(stop)   250                            ;# time to stop
simulation

set num_wired_nodes      2
#set num_bs_nodes       2  ;#this is not really used here.


set ns [new Simulator]

# set up for hierarchical routing
$ns node-config -addressType hierarchical

AddrParams set domain_num_ 3           ;# number of domains
lappend cluster_num 2 1 1              ;# number of clusters in each
domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 2 1            ;# number of nodes in each
cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
#   2 for HA and FA
create-god [expr $opt(nn) + 2]
#====================================
#        Nodes Definition
#====================================
#Create 2 wired nodes
set n0 [$ns node 0.0.0]
set n1 [$ns node 0.1.0]

#====================================
#        Links Definition
#====================================

# Configure for ForeignAgent and HomeAgent nodes
$ns node-config -mobileIP ON \
                 -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                            -topoInstance $topo \
                 -wiredRouting ON \
                            -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON

#create BS0 and BS1 as home agent and foriegn agent
set HA [$ns node 1.0.0]
set FA [$ns node 2.0.0]

# Position (fixed) for base-station nodes (HA & FA).
$HA set X_ 1.000000000000
$HA set Y_ 2.000000000000
$HA set Z_ 0.000000000000

$FA set X_ 650.000000000000
$FA set Y_ 600.000000000000
$FA set Z_ 0.000000000000




#Mobile node can't perform the routing so turn it off
$ns node-config -wiredRouting OFF

# create a mobilenode that would be moving between HA and FA.
# note address of MH indicates its in the same domain as HA.
set MH [$ns node 1.0.1]
set n3 $MH
set HAaddress [AddrParams addr2id [$HA node-addr]]
[$MH set regagent_] set home_agent_ $HAaddress

# movement of the MH
$MH set Z_ 0.000000000000
$MH set Y_ 2.000000000000
$MH set X_ 2.000000000000
$ns at 100.000000000000 "$MH setdest 500.000000000000 500.000000000000
20.000000000000"
# goes back to HA
$ns at 200.000000000000 "$MH setdest 620.000000000000 650.000000000000
20.000000000000"

#links between the wired nodes
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n1 $HA 100.0Mb 10ms DropTail
$ns queue-limit $n1 $HA 50
$ns duplex-link $n1 $FA 100.0Mb 10ms DropTail
$ns queue-limit $n1 $FA 50
```

```
#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $HA orient left-down
$ns duplex-link-op $n1 $FA orient right-down
#====================================
#        Agents Definition
#====================================
set tcp1 [new Agent/TCP]
set sink1 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp1
$ns attach-agent $MH $sink1
$ns connect $tcp1 $sink1
#====================================
#        Applications Definition
#====================================
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"









#====================================
#        Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
#scheduling of ftp
$ns at 180 "$ftp1 stop"
$ns at 200 "finish"

$ns run
```

Q5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.

```
# This script is created by NSG2 beta1
# <http://wushoupong.googlepages.com/nsg>

#===================================
#     Simulation parameters setup
#===================================
Phy/WirelessPhy set freq_ 2.472e9          ;#channel
Phy/WirelessPhy set bandwidth_ 11Mb        ;#Data Rate
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/Tdma                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     7                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      1145                      ;# X dimension of topography
set val(y)      100                      ;# Y dimension of topography


#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
```

```
#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
Phy/WirelessPhy set freq_ 2.472e6

#====================================
#     Mobile node parameter setup
#====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#====================================
#        Nodes Definition
#====================================
#Create 7 nodes
set n0 [$ns node]
$n0 set X_ 250
$n0 set Y_ 133
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 362
$n1 set Y_ 308
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 511
$n2 set Y_ 473
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 680
$n3 set Y_ 345
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 749
$n4 set Y_ 154
$n4 set Z_ 0.0
```

```
$ns initial_node_pos $n4 20

Phy/WirelessPhy set freq_ 2.472e9
$ns node-config -phyType $val(netif)




set n5 [$ns node]
$n5 set X_ 400
$n5 set Y_ 160
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 836
$n6 set Y_ 267
$n6 set Z_ 0.0
$ns initial_node_pos $n6 2

#====================================
#        Agents Definition
#====================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500

#Setup a TCP connection
set tcp2 [new Agent/TCP]
$ns attach-agent $n5 $tcp2
set sink3 [new Agent/TCPSink]
$ns attach-agent $n6 $sink3
$ns connect $tcp2 $sink3
$tcp2 set packetSize_ 1500

#====================================
#        Applications Definition
#====================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 20.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp2
$ns at 1.0 "$ftp1 start"
$ns at 20.0 "$ftp1 stop"
```

```
#====================================
#        Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at 50 "\$n$i reset"
}
$ns at 50 "$ns nam-end-wireless 50"
$ns at 50 "finish"
$ns run
```

PART-B

Q7. Write a program for error detecting code using CRC-CCITT (16- bits).

```java
import java.io.*;
class Crc
{
public static void main(String args[]) throws IOException
{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
int[ ] data;
int[ ]div;
int[ ]divisor;
int[ ]rem;
int[ ] crc;
int data_bits, divisor_bits, tot_length;
System.out.println("Enter number of data bits : ");
data_bits=Integer.parseInt(br.readLine());
data=new int[data_bits];
System.out.println("Enter data bits : ");
for(int i=0; i<data_bits; i++)
data[i]=Integer.parseInt(br.readLine());
System.out.println("Enter number of bits in divisor : ");
divisor_bits=Integer.parseInt(br.readLine());
divisor=new int[divisor_bits];
System.out.println("Enter Divisor bits : ");
for(int i=0; i<divisor_bits; i++)
divisor[i]=Integer.parseInt(br.readLine());
/*
System.out.print("Data bits are : ");
for(int i=0; i< data_bits; i++)
System.out.print(data[i]);
System.out.println();
System.out.print("divisor bits are : ");
for(int i=0; i< divisor_bits; i++)
System.out.print(divisor[i]);
System.out.println();
*/
tot_length=data_bits+divisor_bits-1;
div=new int[tot_length];
rem=new int[tot_length];
crc=new int[tot_length];
/*------------------ CRC GENERATION-----------------------*/
for(int i=0;i<data.length;i++)
div[i]=data[i];
System.out.print("Dividend (after appending 0's) are : ");
for(int i=0; i< div.length; i++)
System.out.print(div[i]);
System.out.println();
for(int j=0; j<div.length; j++){
rem[j] = div[j];
}


rem=divide(div, divisor, rem);
for(int i=0;i<div.length;i++)
//append dividend and ramainder
{
```

```
crc[i]=(div[i]^rem[i]);
}
System.out.println();
System.out.println("CRC code : ");
for(int i=0;i<crc.length;i++)
System.out.print(crc[i]);
/*-------------------ERROR DETECTION---------------------*/
System.out.println();
System.out.println("Enter CRC code of "+tot_length+" bits : ");
for(int i=0; i<crc.length; i++)
crc[i]=Integer.parseInt(br.readLine());
/*
System.out.print("crc bits are : ");
for(int i=0; i< crc.length; i++)
System.out.print(crc[i]);
System.out.println();
*/
for(int j=0; j<crc.length; j++){
rem[j] = crc[j];
}
rem=divide(crc, divisor, rem);
for(int i=0; i< rem.length; i++)
{
if(rem[i]!=0)
{
System.out.println("Error");
break;
}
if(i==rem.length-1)
System.out.println("No Error");
}
System.out.println("THANK YOU.... :)");
}
static int[] divide(int div[],int divisor[], int rem[])
{
int cur=0;
while(true)
{
for(int i=0;i<divisor.length;i++)
rem[cur+i]=(rem[cur+i]^divisor[i]);
while(rem[cur]==0 && cur!=rem.length-1)
cur++;
if((rem.length-cur)<divisor.length)
break;
}
return rem;
}
}
```

<u>OUTPUT:-</u>

```
student@unixlab:~$ java Crc
Enter number of data bits :
```

```
4
Enter data bits :
1
1
0
1
Enter number of bits in divisor :
17
Enter Divisor bits :
1
0
0
0
1
0
0
0
0
0
0
1
0
0
0
0
1
```

```
Dividend (after appending 0's) are : 11010000000000000000
CRC code :
11011101000110101101
Enter CRC code of 20 bits :
1
1
0
```

```
1
1
1
0
1
0
0
0
1
1
0
1
0
1
1
0
1
No Error
THANK YOU.... :)
```

Q8. Write a program to find the shortest path between vertices using bellman-ford algorithm.

```
import java.util.Scanner;
public class BellmanFord
{
private int D[];
private int num_ver;
public static final int MAX_VALUE = 999;
public BellmanFord(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();
BellmanFord b = new BellmanFord (num_ver);
b.BellmanFordEvaluation(source, A);
scanner.close();
}
}
```

<u>OUTPUT:-</u>

```
student@unixlab:~$ javac BellmanFord.java
student@unixlab:~$ java BellmanFord
Enter the number of vertices
4
Enter the adjacency matrix
0 5 0 0
5 0 3 4
0 3 0 2
0 4 2 0
Enter the source vertex
2
distance of source 2 to 1 is 5
distance of source 2 to 2 is 0
distance of source 2 to 3 is 3
distance of source 2 to 4 is 4
```

Q9.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.

Server Side:-

```
import java.net.*;
import java.io.*;
public class ContentsServer
{
  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 wating 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();
  }
}
```

Client Side:-

```
import java.net.*;
import java.io.*;
public class ContentsClient
{
  public static void main( String args[ ] ) throws Exception
  {
     Socket sock = new Socket( "127.0.0.1", 4000);
     System.out.print("Enter the file name");
     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:-

```
student@unixlab:~$ javac ContentsServer.java
student@unixlab:~$ java ContentsServer
Server ready for connection
Connection is successful and wating for chatting
```

```
student@unixlab:~$ javac ContentsClient.java
student@unixlab:~$ java ContentsClient
Enter the file name
abc.txt
TCP is Reliable Protocol
```

Q10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.

Server Side:-

```
import java.io.*;
import java.net.*;
class UDPServer {
public static void main(String args[]) throws Exception
{
DatagramSocket serverSocket = new DatagramSocket(6789);
byte[] receiveData = new byte[1024];
byte[] sendData = new byte[1024];
while(true)
{
DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
serverSocket.receive(receivePacket);
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();

System.out.println("enter message to echo:");
BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
String Sentence = inFromUser.readLine();
sendData = Sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress,port);
serverSocket.send(sendPacket);

}
}
}
```

Client Side:-

```
import java.io.*;
import java.net.*;
class UDPClient {
public static void main(String args[]) throws Exception
{
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName("localhost");
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
String sentence = "hi";
sendData = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress,6789);
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:-

```
student@unixlab:~$ javac UDPServer.java
student@unixlab:~$ java UDPServer
enter message to echo:
TCP is Reliable Protocol
```

```
student@unixlab:~$ javac UDPClient.java
student@unixlab:~$ java UDPClient
FROM SERVER:TCP is Reliable Protocol
```

Q11. Write a program for simple RSA algorithm to encrypt and decrypt the data.

RSA Key Generation

```
import java.util.*;
import java.math.BigInteger;
import java.lang.*;
class RSAKeygen
{
public static void main(String[] args)
{
Random r1=new Random(System.currentTimeMillis());
Random r2=new Random(System.currentTimeMillis()*10);
int e=Integer.parseInt(args[0]);
BigInteger p=BigInteger.probablePrime(32, r1);
BigInteger q=BigInteger.probablePrime(32, r2);
BigInteger n=p.multiply(q);
BigInteger p1=p.subtract(new BigInteger("1"));
BigInteger q1=q.subtract(new BigInteger("1"));
BigInteger phi=p1.multiply(q1);
while(true)
{
BigInteger GCD1=phi.gcd(new BigInteger(""+e));
if(GCD1.equals(BigInteger.ONE))
{
break;
}
e++;
}
BigInteger pubkey=new BigInteger(""+e);
BigInteger prvkey=pubkey.modInverse(phi);
System.out.println("public key : "+pubkey+","+n);
System.out.println("private key : "+prvkey+","+n);
}
}
```

RSA Encryption and Decryption

```
import java.math.BigInteger;
import java.util.*;
class RSAEncDec
{
public static void main(String[] args)
{
BigInteger pubkey = new BigInteger(args[0]);
BigInteger prvkey = new BigInteger(args[1]);
BigInteger n = new BigInteger(args[2]);
int m=Integer.parseInt(args[3]);
BigInteger val=new BigInteger(""+m);
BigInteger cipher=val.modPow(pubkey,n);
System.out.println("Cipher text: " + cipher);
BigInteger plain=cipher.modPow(prvkey,n);
int plainVal=plain.intValue();
System.out.println("Plain text:" + plainVal);
}
}
```

<u>OUTPUT:-</u>

```
student@unixlab:~$ javac RSAKeygen.java
student@unixlab:~$ java RSAKeygen 20
public key : 23,5986689899622219251
private key : 2602908649882022327,5986689899622219251
student@unixlab:~$ java RSAEncDec 23 2602908649882022327
5986689899622219251 97
Cipher text: 1144628385513821454
Plain text:97
```

Q12. Write a program for congestion control using leaky bucket algorithm.

```
import java.util.*;
public class leaky
{
    public static void main(String[] args)
    {
        Scanner my = new Scanner(System.in);
        int no_groups,bucket_size;
        System.out.print("\n Enter the bucket size : \t");
        bucket_size = my.nextInt();
        System.out.print("\n Enter the no of groups : \t");
        no_groups = my.nextInt();
        int no_packets[] = new int[no_groups];
        int in_bw[] = new int[no_groups];
        int out_bw,reqd_bw=0,tot_packets=0;
        for(int i=0;i<no_groups;i++)
        {
            System.out.print("\n Enter the no of packets for group " +
(i+1) + "\t");
            no_packets[i] = my.nextInt();
            System.out.print("\n Enter the input bandwidth for the
group " + (i+1) + "\t");
            in_bw[i] = my.nextInt();
            if((tot_packets+no_packets[i])<=bucket_size)
            {
                tot_packets += no_packets[i];
            }
            else
            {
                do
                {
                System.out.println(" Bucket Overflow ");
                System.out.println(" Enter value less than " +
(bucket_size-tot_packets));
                no_packets[i] = my.nextInt();
                }while((tot_packets+no_packets[i])>bucket_size);
                tot_packets += no_packets[i];
            }
            reqd_bw += (no_packets[i]*in_bw[i]);
        }  System.out.println("\nThe total required bandwidth is " +
reqd_bw);
        System.out.println("Enter the output bandwidth ");
        out_bw = my.nextInt();
        int temp=reqd_bw;
        int rem_pkts = tot_packets;




        while((out_bw<=temp)&&(rem_pkts>0))
        {
            System.out.println("Data Sent \n" + (--rem_pkts) + " 
packets remaining");
            System.out.println("Remaining Bandwidth " + (temp -= 
out_bw));
            if((out_bw>temp)&&(rem_pkts>0))
                System.out.println(rem_pkts + " packet(s) discarded 
due to insufficient bandwidth");
        }
    }
}
```

OUTPUT:-

```
student@unixlab:~$ gedit leaky.java
student@unixlab:~$ javac leaky.java
student@unixlab:~$ java leaky
Enter the bucket size :              10

Enter the no of groups :              2

Enter the no of packets for group 1              5

Enter the input bandwidth for the group 1              3

Enter the no of packets for group 2              5

Enter the input bandwidth for the group 2              3

The total required bandwidth is 30
Enter the output bandwidth
2
Data Sent
9 packets remaining
Remaining Bandwidth 28
Data Sent
8 packets remaining
Remaining Bandwidth 26
Data Sent
7 packets remaining
Remaining Bandwidth 24
Data Sent
6 packets remaining
Remaining Bandwidth 22
Data Sent
5 packets remaining
Remaining Bandwidth 20
Data Sent
```

```
4 packets remaining
Remaining Bandwidth 18
Data Sent
3 packets remaining
Remaining Bandwidth 16
Data Sent
2 packets remaining
Remaining Bandwidth 14
Data Sent
1 packets remaining
Remaining Bandwidth 12
Data Sent
0 packets remaining
Remaining Bandwidth 10

student@unixlab:~$ java leaky
Enter the bucket size :               10

Enter the no of groups :               2

Enter the no of packets for group 1              5

Enter the input bandwidth for the group 1              3

Enter the no of packets for group 2              6

Enter the input bandwidth for the group 2              3
Bucket Overflow
Enter value less than 5
5
The total required bandwidth is 30
Enter the output bandwidth
4
Data Sent
9 packets remaining
Remaining Bandwidth 26
Data Sent
8 packets remaining
Remaining Bandwidth 22
Data Sent
7 packets remaining
Remaining Bandwidth 18
Data Sent
6 packets remaining
Remaining Bandwidth 14
Data Sent
5 packets remaining
Remaining Bandwidth 10
Data Sent
4 packets remaining
Remaining Bandwidth 6
Data Sent
3 packets remaining
Remaining Bandwidth 2
3 packet(s) discarded due to insufficient bandwidth
```