CN LAB MANUAL

<u>ATTENTION! ATTENTION! ATTENTION!</u>

PLEASE CHANGE PART OF CODE AND INPUT FOR GETTING OUTPUT ACCORDING TO YOUR FILE NAME

THE CODE IS RUN AND VERIFIED, EVEN THOUGH WE DON'T TAKE ANY RESPONSIBILITIES!

PART A

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.

```
set ns [new Simulator]
$ns color 1 Red
$ns color 2 Blue
set nf [open l1.nam w]
$ns namtrace-all $nf
set nt [open l1.tr w]
$ns trace-all $nt
proc finish {} {
global ns nf nt
$ns flush-trace
close $nf
close $nt
exec nam l1.nam &
exec cat l1.tr | awk -f l1.awk &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label "source1"
$n1 label "source2"
```

```
    $ns duplex-link $n0 $n2 1Mb 15ms DropTail
    $ns duplex-link $n1 $n2 1Mb 15ms DropTail
    $ns duplex-link-op $n0 $n2 orient right-down
    $ns duplex-link-op $n1 $n2 orient right-up
    set udp0 [new Agent/UDP]
    $ns attach-agent $n0 $udp0
    set udp1 [new Agent/UDP]
    $ns attach-agent $n1 $udp1
    set null0 [new Agent/Null]
    $ns attach-agent $n2 $null0
    $ns set queue-limit $n0 $n2 10
    $ns set queue-limit $n1 $n2 5
    set cbr0 [new Application/Traffic/CBR]
    $cbr0 set packet-size_ 600
    $cbr0 set interval_ 0.0002
    $cbr0 attach-agent $udp0
    set cbr1 [new Application/Traffic/CBR]
    $cbr1 set packet-size_ 1000
    $cbr1 set interval_ 0.002
    $cbr1 attach-agent $udp1
    $ns connect $udp0 $null0
    $udp0 set fid_ 1
    $ns connect $udp1 $null0
    $udp1 set fid_ 2
    $ns at 0.1 "$cbr0 start"
    $ns at 0.2 "$cbr1 start"
    $ns at 2.5 "$cbr0 stop"
    $ns at 2.8 "$cbr1 stop"
    $ns at 3.0 "finish"
    $ns run
BEGIN{
```

```
count=0;
}
{
if($1=="d")
count++;
}
END{
printf("Dropped packets are:%d\n",count);
}
```

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.

```
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 [$ns node]
set n6 [$ns node]
$n0 label "ping0"
$n4 label "ping4"
$n5 label "ping5"
$n6 label "ping6"
$n2 label "Router"
$ns color 1 "red"
$ns color 2 "green"
$ns duplex-link $n0 $n2 100Mb 300ms DropTail
$ns duplex-link $n2 $n6 1Mb 300ms DropTail
$ns duplex-link $n5 $n2 100Mb 300ms DropTail
$ns duplex-link $n2 $n4 1Mb 300ms DropTail
$ns duplex-link $n3 $n2 1Mb 300ms DropTail
$ns duplex-link $n1 $n2 1Mb 300ms DropTail
$ns queue-limit $n0 $n2 5
$ns queue-limit $n2 $n6 2
$ns queue-limit $n2 $n4 3
$ns queue-limit $n5 $n2 5
set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0
set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4
set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5
set ping6 [new Agent/Ping]
$ns attach-agent $n6 $ping6
$ping0 set packetSize_ 50000
$ping0 set interval_ 0.0001
$ping5 set packetSize_ 60000
$ping5 set interval_ 0.0001
```

```
$ping0 set class_ 1
$ping5 set class_ 2
$ns connect $ping0 $ping4
$ns connect $ping5 $ping6
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "the node [$node_ id]received an reply from $from
with round trip time to $rtt"}
proc finish {} {
global ns nf tf
exec nam lab2.nam &
exec cat lab2.nam | awk -f prog2.awk &
$ns flush-trace
close $tf
close $nf
exit 0
}
$ns rtmodel-at 0.9 down $n2 $n6
$ns rtmodel-at 1.5 up $n2 $n6
$ns at 0.1 "$ping0 send"
$ns at 0.2 "$ping0 send"
$ns at 0.3 "$ping0 send"
$ns at 0.4 "$ping0 send"
$ns at 0.5 "$ping0 send"
$ns at 0.6 "$ping0 send"
$ns at 0.7 "$ping0 send"
$ns at 0.8 "$ping0 send"
$ns at 0.9 "$ping0 send"
$ns at 1.0 "$ping0 send"
$ns at 1.1 "$ping0 send"
$ns at 1.2 "$ping0 send"
$ns at 1.3 "$ping0 send"
$ns at 1.4 "$ping0 send"
$ns at 1.5 "$ping0 send"
$ns at 1.6 "$ping0 send"
$ns at 1.7 "$ping0 send"
$ns at 1.8 "$ping0 send"
$ns at 0.1 "$ping5 send"
$ns at 0.2 "$ping5 send"
$ns at 0.3 "$ping5 send"
$ns at 0.4 "$ping5 send"
$ns at 0.5 "$ping5 send"
$ns at 0.6 "$ping5 send"
$ns at 0.7 "$ping5 send"
$ns at 0.8 "$ping5 send"
$ns at 0.9 "$ping5 send"
$ns at 1.0 "$ping5 send"
$ns at 1.1 "$ping5 send"
$ns at 1.2 "$ping5 send"
$ns at 1.3 "$ping5 send"
$ns at 1.4 "$ping5 send"
$ns at 1.5 "$ping5 send"
$ns at 1.6 "$ping5 send"
$ns at 1.7 "$ping5 send"
$ns at 1.8 "$ping5 send"
$ns at 5.0 "finish"
```

```
$ns run

BEGIN {
#include<stdio.h>
count=0;
}
{ if($1=="d")
count++;
}
END {
printf("the total no of packets Dropped due to
congestion:%d",count);
}
```

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

```
set ns [new Simulator]

$ns color 1 Red

$ns color 2 Blue

set tf [open pr3.tr w]

$ns trace-all $tf

set nf [open pr3.nam w]

$ns namtrace-all $nf

proc finish {} {

        global nf ns tf

        exec nam pr3.nam &

        exec cat pr3.nam | awk -f pr3.awk &

        $ns flush-trace

        close $nf

        close $tf

        exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]
```

```
$ns make-lan "$n0 $n1 $n2 $n3" 10Mb 10ms LL
Queue/DropTail Mac/802_3
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n1 $sink2
$ns connect $tcp2 $sink2
set file1 [open file1.tr w]
$tcp0 attach $file1
$tcp0 trace cwnd_
$tcp0 set maxcwnd_ 10
set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp2 trace cwnd_
$tcp0 set fid_ 1
$tcp2 set fid_ 2
$ns at 0.1 "$ftp0 start"
$ns at 1.5 "$ftp0 stop"
$ns at 2 "$ftp0 start"
$ns at 3 "$ftp0 stop"
$ns at 0.2 "$ftp2 start"
$ns at 2 "$ftp2 stop"
```

```
$ns at 2.5 "$ftp2 start"

$ns at 4 "$ftp2 stop"

$ns at 5 "finish"

$ns run
```

```
BEGIN {
        #include<stdio.h>
      }
      {
        if($6=="cwnd_")
          printf("%f\t%f\n",$1,$7);
     }
END {
      puts "DONE"
}
```

Graph commands

```
awk -f pr3.awk file1.tr > tcp1
awk -f pr3.awk file2.tr > tcp2
xgraph -x “time” y “cwnd” tcp1 tcp2
```

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

```
set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(x) 1000

set val(y) 1000

set val(ifqlen) 50

set val(nn) 3

set val(stop) 250.0

set val(rp) DSDV
set ns [new Simulator]

set tf [open 4114.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load_flatgrid 1000 1000

set nf [open 4114.nam w]

$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV \

                    -llType LL \

                    -macType Mac/802_11 \

                    -ifqType Queue/DropTail \

                    -ifqLen 50 \

                    -phyType Phy/WirelessPhy \

                    -channelType Channel/WirelessChannel \

                    -propType Propagation/TwoRayGround \

                    -antType Antenna/OmniAntenna \

                    -topoInstance $topo \

                    -agentTrace ON \

                    -routerTrace ON

create-god 3

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

$n0 label "tcp0"

$n1 label "sink1/tcp1"

$n2 label "sink2"

$n0 set X_ 50

$n0 set Y_ 50

$n0 set Z_ 0
```

```
$n1 set X_ 100

$n1 set Y_ 100

$n1 set Z_ 0


$n2 set X_ 600

$n2 set Y_ 600

$n2 set Z_ 0


$ns at 0.1 "$n0 setdest 50 50 15"

$ns at 0.1 "$n1 setdest 100 100 25"

$ns at 0.1 "$n2 setdest 600 600 25"

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0


set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0


set sink1 [new Agent/TCPSink]

$ns attach-agent $n1 $sink1


$ns connect $tcp0 $sink1

set tcp1 [new Agent/TCP]


$ns attach-agent $n1 $tcp1

set ftp1 [new Application/FTP]


$ftp1 attach-agent $tcp1

set sink2 [new Agent/TCPSink]


$ns attach-agent $n2 $sink2

$ns connect $tcp1 $sink2
```

```
$ns at 5 "$ftp0 start"

$ns at 5 "$ftp1 start"

$ns at 100  "$n1  setdest  550 550 15"

$ns at 190 "$n1  setdest  70 70 15"


proc finish {} {

    global ns nf tf

    $ns flush-trace

    exec nam 4114.nam &

    exec cat 4114.tr | awk -f 4114.awk &

    close $tf

    close $nf

   exit 0

   }

$ns at 250 "finish"

$ns run
```

```
BEGIN{
     #include<stdio.h>
      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("the throughput from no to
n1:%fmbps\n",((count1*pack1*8)/(time1*1000000)));
printf("the throughput from n1 to
n2:%fmbps\n",((count2*pack2*8)/(time2*1000000)));
}
```

NOTE: Directory to save and execute 5th and 6th pgms
Change directory using following command in terminal
cd /home/cs/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts

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

```
set stop 100    ;    # Stop time.
# Topology
set type gsm    ;#type of link:
# AQM parameters
set minth 30    ;
set maxth 0     ;
set adaptive 1  ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set flows 0     ;# number of long-lived TCP flows
set window 30   ;# window for long-lived traffic
set web 2       ;# number of web sessions
# Plotting statics.
set opt(wrap)    100  ;# wrap plots?
set opt(srcTrace) is  ;# where to plot traffic
set opt(dstTrace) bs2 ;# where to plot traffic

#default downlink bandwidth in bps
set bwDL(gsm)  9600
#default uplink bandwidth in bps
set bwUL(gsm)  9600
#default downlink propagation delay in seconds
set propDL(gsm)  .500
#default uplink propagation delay in seconds
set propUL(gsm)  .500

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]
proc cell_topo {} {
  global ns nodes
  $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10nodes(ms) DropTail
  $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
  $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
  $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50nodes(ms) DropTail
  puts " GSM Cell Topology"
```

```
}
proc set_link_para {t} {
  global ns nodes bwUL bwDL propUL propDL buf
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) duplex
  $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) duplex
  $ns delay $nodes(bs1) $nodes(ms) $propDL($t) duplex
  $ns delay $nodes(bs2) $nodes(ms) $propDL($t) duplex
  $ns queue-limit $nodes(bs1) $nodes(ms) 10
  $ns queue-limit $nodes(bs2) $nodes(ms) 10
}
# RED and TCP parameters
Queue/RED set adaptive_ $adaptive
Queue/RED set thresh_ $minth
Queue/RED set maxthresh_ $maxth
Agent/TCP set window_ $window
source web.tcl

#Create topology
switch $type {
 gsm -
 gprs -
 umts {cell_topo}
}
  set_link_para $type
  $ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
  $ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

# Set up forward TCP connection
if {$flows == 0} {
      set tcp1 [$ns create-connection TCP/Sack1 $nodes(is)
TCPSink/Sack1 $nodes(lp) 0]
      set ftp1 [[set tcp1] attach-app FTP]
      $ns at 0.8 "[set ftp1] start"
}

if {$flows > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is)
TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0  "[set ftp1] start"
$ns at 3.5  "[set ftp1] stop"
    set tcp2 [$ns create-connection TCP/Sack1 $nodes(is)
TCPSink/Sack1 $nodes(lp) 0]
    set ftp2 [[set tcp2] attach-app FTP]
    $tcp2 set window_ 3
    $ns at 1.0  "[set ftp2] start"
    $ns at 8.0  "[set ftp2] stop"
}

proc stop {} {
      global nodes opt nf
         set wrap $opt(wrap)
      set sid [$nodes($opt(srcTrace)) id]
      set did [$nodes($opt(dstTrace)) id]
      set a "out.tr"
```

```
     set GETRC "../../../bin/getrc"
        set RAW2XG "../../../bin/raw2xg"
        exec $GETRC -s $sid -d $did -f 0 out.tr | \
          $RAW2XG -s 0.01  -m $wrap -r > plot.xgr
        exec $GETRC -s $did -d $sid -f 0 out.tr | \
          $RAW2XG -a -s 0.01 -m $wrap  >> plot.xgr
        exec xgraph -x time -y packets plot.xgr &
     exit 0
}
$ns at $stop "stop"
$ns run
```

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

```
set stop 100          ;     # Stop time.
# Topology
set type cdma    ;     #type of link:
# AQM parameters
set minth 30          ;
set maxth 0           ;
set adaptive 1   ;     # 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set flows 0      ;          # number of long-lived TCP flows
set window 30    ;     # window for long-lived traffic
set web 2        ;     # number of web sessions
# Plotting statics.
set opt(wrap)    100  ;     # wrap plots?
set opt(srcTrace) is        ;     # where to plot traffic
set opt(dstTrace) bs2 ;     # where to plot traffic

#default downlink bandwidth in bps
set bwDL(cdma) 384000
#default uplink bandwidth in bps
set bwUL(cdma) 64000
#default downlink propagation delay in seconds
set propDL(cdma) .150
#default uplink propagation delay in seconds
set propUL(cdma) .150

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]


proc cell_topo {} {
  global ns nodes
  $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10nodes(ms) DropTail
  $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
  $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
  $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50nodes(ms) DropTail
  puts " cdma Cell Topology"
}
proc set_link_para {t} {
  global ns nodes bwUL bwDL propUL propDL buf
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) duplex
 $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) duplex
  $ns delay $nodes(bs1) $nodes(ms) $propDL($t) duplex
  $ns delay $nodes(bs2) $nodes(ms) $propDL($t) duplex
  $ns queue-limit $nodes(bs1) $nodes(ms) 20
  $ns queue-limit $nodes(bs2) $nodes(ms) 20
}
# RED and TCP parameters
Queue/RED set adaptive_ $adaptive
Queue/RED set thresh_ $minth
Queue/RED set maxthresh_ $maxth
Agent/TCP set window_ $window

source web.tcl

#Create topology
switch $type {

cdma {cell_topo}
}
  set_link_para $type
  $ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
  $ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

# Set up forward TCP connection
if {$flows == 0} {
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1
$nodes(lp) 0]
set ftp1 [[set tcp1] attach-app FTP]
$ns at 0.8 "[set ftp1] start"
}
if {$flows > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is)
TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0  "[set ftp1] start"
    $ns at 3.5  "[set ftp1] stop"
    set tcp2 [$ns create-connection TCP/Sack1 $nodes(is)
TCPSink/Sack1 $nodes(lp) 0]
    set ftp2 [[set tcp2] attach-app FTP]
    $tcp2 set window_ 3
    $ns at 1.0  "[set ftp2] start"
    $ns at 8.0  "[set ftp2] stop"
}

proc stop {} {
global nodes opt nf
           set wrap $opt(wrap)
      set sid [$nodes($opt(srcTrace)) id]
set did [$nodes($opt(dstTrace)) id]
      set a "out.tr"
      set GETRC "../../../bin/getrc"
           set RAW2XG "../../../bin/raw2xg"
      exec $GETRC -s $sid -d $did -f 0 out.tr | \
          $RAW2XG -s 0.01  -m $wrap -r > plot.xgr
           exec $GETRC -s $did -d $sid -f 0 out.tr | \
          $RAW2XG -a -s 0.01 -m $wrap  >> plot.xgr
           exec xgraph -x time -y packets plot.xgr &
      exit 0
}
$ns at $stop "stop"
$ns run
```

PART B

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

```
import java.util.Scanner;

class CRC1

{

  public static void main(String args[])

  {

      Scanner sc =new Scanner(System.in);

      System.out.println("enter data stream");

      String datastream=sc.nextLine();

      System.out.println("enter generator:");

      String generator=sc.nextLine();

      int data[]=new int[datastream.length()+generator.length()-1];

      int divisor[]=new int[generator.length()];

      for(int i=0;i<datastream.length();i++)

      data[i]=Integer.parseInt(datastream.charAt(i)+ "");
```

```
     for(int i=0;i<generator.length();i++)
     divisor[i]=Integer.parseInt(generator.charAt(i)+"");

    for(int i=0;i<datastream.length();i++)
     {
       if(data[i]==1)
       for(int j=0;j<divisor.length;j++)
       data[i+j]^=divisor[j];
     }

    System.out.println("the CRC code is:");
    for(int i=0;i<datastream.length();i++)
    data[i]=Integer.parseInt(datastream.charAt(i)+"");
    for(int i=0;i<data.length;i++)
    System.out.print(data[i]);
    System.out.println();

   System.out.print("enter CRC code:");
   datastream=sc.nextLine();
   System.out.println("enter generator:");
   generator=sc.nextLine();
   data=new int[datastream.length()+generator.length() -1];
   divisor=new int[generator.length()];
   for(int i=0;i<datastream.length();i++)
   data[i]=Integer.parseInt(datastream.charAt(i)+"");
   for(int i=0;i<generator.length();i++)
   divisor[i]=Integer.parseInt(generator.charAt(i)+"");

   for(int i=0;i<datastream.length();i++)
    {
     if(data[i]==1)
```

```
        for(int j=0;j<divisor.length;j++)
        data[i+j]^=divisor[j];
       }

      boolean valid=true;
      for(int i=0;i<data.length;i++)
       if(data[i]==1)
       {
         valid=false;
         break;
        }
        if(valid==true)
        System.out.println("data stream is valid");
         else
       System.out.println("data stream is invalid CRC error occured");
       }
      }
```

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

```
import java.util.Scanner;
public class B
{
private int D[];
private int num;
public static final int Max=999;

public B(int num)
{
this.num=num;
D=new int [num+1];
}
```

```
public void BE(int s, int A[][])

{


        for(int node=1; node<=num;node++)

{

D[node]=Max;

}

D[s]=0;


for(int node=1;node<=num-1;node++)

{

for(int sn=1; sn<=num;sn++)

{

for(int dn=1;dn<=num;dn++)

{

if(A[sn][dn]!=Max)

{

if(D[dn]>D[sn]+A[sn][dn])

D[dn]=D[sn]+A[sn][dn];

}

}

}

}


for(int sn=1; sn<=num;sn++)

{

for(int dn=1;dn<=num;dn++)

{

if(A[sn][dn]!=Max)

{

if(D[dn]>D[sn]+A[sn][dn])
System.out.println("Negative cycle");
}
}
}
for(int v=1;v<=num;v++)
{
System.out.println("Distance of source"+ s+ "to " + v+ "is"+ D[v]);
}
}
public static void main (String args[])
{
int num=0; int s;
Scanner input=new Scanner (System.in);
System.out.println("Enetr vertex");
num=input.nextInt();
int A[][]=new int[num+1][num+1];
System.out.println("enetr matrix");
for( int sn=1;sn<=num;sn++)
{
for (int dn=1;dn<=num;dn++)
{
A[sn][dn]=input.nextInt();
if(sn==dn)
{
A[sn][dn]=0; continue;
}
if(A[sn][dn]==0)
{
A[sn][dn]=Max;
}
}
```

```
}
System.out.println("Eneter source vertex");
s=input.nextInt();
B bb=new B(num);
bb.BE(s,A);
input.close();
}
}
```

9. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

```
//client side


import java.net.*;
import java.io.*;
 public class tcpclient
{
public static void main(String args[]) throws Exception
{
Socket sock=new Socket ("127.00.1",8080);
System.out.println("cannection acc with server");
System.out.println("Enetr file name");
BufferedReader keyRead=new BufferedReader (new InputStreamReader(System.in));
String fname=keyRead.readLine();
File f=new File(fname);
OutputStream ostream=sock.getOutputStream();
PrintWriter pwrite =new PrintWriter(ostream,true);
pwrite.println(fname);
if(!f.exists()||f.isDirectory())
{
System.out.println("File doesnt exist");
System.exit(0);
}
InputStream istream =sock.getInputStream();
BufferedReader socketRead=new BufferedReader(new InputStreamReader(istream));
System.out.println("Server acc the req & sending\n");
System.out.println("Contents of req file are  ");
System.out.println("---------");
String str;
while((str=socketRead.readLine())!=null)
{
System.out.println(str);
}
pwrite.close();
sock.close();
}
}
//server side
import java.net.*;
import java.io.*;
public class tcpserver
{
public static void main (String args[]) throws Exception
{
ServerSocket sersock=new ServerSocket (8080);
System.out.println("Server ready for connectrion");
Socket sock=sersock.accept();
System.out.println("Connection success waut client");
InputStream istream =sock.getInputStream();
BufferedReader fileRead=new BufferedReader(new InputStreamReader(istream));
String fname=fileRead.readLine();
File f=new File(fname);
```

```
if(!f.exists())
{
System.out.println("Client req file  not existr");
System.exit(0);
}

System.out.println("A req for filename"+" " + fname+ "  " + " is recieved");
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);
}
System.out.println("Req closed");
sersock.close();
pwrite.close();
fileRead.close();
contentRead.close();
}
}
```

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

```
//client side

import java.net.*;
public class DReciever
{
	public static void main(String[] args) throws Exception
	{
		byte[] buf = new byte[1024];
		System.out.println("Receiver");
		DatagramSocket ds = new DatagramSocket(3000);

		while(true)
		{
			DatagramPacket dp = new DatagramPacket(buf, 1024);
			ds.receive(dp);
			String msg = new String(dp.getData(), 0, dp.getLength());
			System.out.println(msg);
		}
	}
}
//server side
import java.net.*;
import java.util.Scanner;
public class DSender
{
	public static void main(String[] args) throws Exception
	{
		System.out.println("Sender");
		DatagramSocket ds = new DatagramSocket();
		Scanner scanner = new Scanner(System.in);

		System.out.println("\nEnter the Message : ");

		while(true)
		{
			String msg = scanner.nextLine();
```

```
                    InetAddress ip = InetAddress.getByName("127.0.0.1");

                DatagramPacket dp = new DatagramPacket(msg.getBytes(), msg.length(), ip, 3000);
                    ds.send(dp);
                }
            }
}
```

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

```
import java.util.Scanner;
public class RRSA
{
public static int mult(int x,int y,int n)
{
int k=1;
int j;
for(j=1;j<=y;j++)
k=(k*x)%n;
return k;
}
public static int gcd(int m,int n)
{
if(n==0)
return m;
else
return(gcd(n,m%n));
}
public static void main(String[] args)
{
int msg,plainText,cipherText;
int n,d=0,e,z,p,q,i;
Scanner sc=new Scanner(System.in);
System.out.println("enter the value of primes p and q:");
p=sc.nextInt();
q=sc.nextInt();
System.out.println("enter message such that (message less than or equal to ((p*q)-2):");
msg=sc.nextInt();
n=p*q;
z=(p-1)*(q-1);
do
{
System.out.println("choose the value of e (e>2) such that gcd(z,e)=1:");
e=sc.nextInt();
} while(gcd(z,e)!=1);
i=2;
while(((i*e)%z)!=1)
{
i++;
d=i;
}
System.out.println("the public key pair is ("+e+","+n+")");
System.out.println("the private key pair is("+d+","+n+")");
cipherText=mult(msg,e,n);
System.out.println("cipher Text="+cipherText);
plainText=mult(cipherText,d,n);
System.out.println("plain Text="+plainText);
}
}
```

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

```
import java.util.Scanner;
public class lleaky
{
public static int bucketSize=1000;
public static int outputRate=100;
public static void sendPacket(int pktSize)
{
if(pktSize>bucketSize)
{
System.out.println("bucket overflow");
}
else
{
while(pktSize>outputRate)
{
System.out.println(outputRate+"bytes of packet is sent");
pktSize=pktSize-outputRate;
}
System.out.println(pktSize+"bytes of packet is sent");
}
}
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("enter the number of packets:");
int numpackets=sc.nextInt();
if(numpackets>0)
{
for(int i=1;i<=numpackets;i++)
{
```

```
System.out.println("enter the packet"+i+"size:");
int pktSize=sc.nextInt();
sendPacket(pktSize);
}
}
else
{
System.out.println("no packets to send");
}
}
}
```