**Lab 8**

Q1

puts "Enter First Number To Check Even Odd: "

set num1 [gets stdin]

for {set i 0} {$i>=0} {incr i} {

if {$num1 <= 4} {

puts "Invalid input. Please enter a number greater than 4."

set num1 [gets stdin]

} else break

}

if {$num1 % 2 == 0} {

puts "This Number Is Even"

set num2 4

for {set i 4} {$i < $num1} {incr i} {

if {$i % 2 == 0}{

puts "i $i"

}

}

} else {

puts "This Number Is Odd"

set num2 3

for {set i 3} {$i < $num1} {incr i} {

if {$i % 2 != 0} {

puts "i $i"

}

}

}

Q2

#This program will create two nodes and pass traffice from one node to other node using TCP protocol

#Creating simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Creating the nam file

set nf [open out.nam w]

#It opens the file 'out.nam' for writing and gives it the file handle 'nf'.

$ns namtrace-all $nf

# Open the trace file

# set nf [open out.tr w]

# $ns trace-all $nf

# we are opening a newtrace file named as "out" and also telling that data must be stored in .tr [trace] format.

# "nf" is the file handler that we are used here to handle the trace file.

# "w" means write i.e the file out.tr is opened for writing.

# "r" means reading

# The second line tells the simulator to trace each packet on every link in the topology and for that we give file handler nf for the simulator ns.

#Finish Procedure (closes the trace file and starts nam)

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}

#The trace data is flushed into the file by using command $ns flush-trace and then file is closed.

#Creating four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Creating a duplex link between two nodes (connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a SFQ queue)

#The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay.

$ns duplex-link $n0 $n2 2Mb 10ms SFQ DropTail

$ns duplex-link $n1 $n2 2Mb 10ms SFQ DropTail

#The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay.

$ns duplex-link $n2 $n3 1.7Mb 20ms SFQ DropTail

#Set Queue size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 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 duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

#Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid\_ 2

#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1mb

$cbr set random\_ false

#Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Print CBR packet size and interval

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

#Run the simulation

$ns run

