COMPUTER NETWORKS Lab

[As per Choice Based Credit System (CBCS) & OBE Scheme]

SEMESTER – VI

Course Code:	**P21ISL506**	**Credits:**	**1**
Teaching Hours/Week (L:T:P):	**0:0:2**	**CIE Marks:**	**50**
Total Theory Teaching Hours: **Total Laboratory Hours:**	**24**	**SEE Marks:**	**50**

Course Learning Objectives: This course will enable the students to:

1. Analyze the performance of various network topologies
2. Design and demonstrate various data flow control and routing algorithms
3. Explore the concept of error detection and routing mechanism
4. Explore various Networking tools

Instructions: Students has to attend one Question from each part in SEE

COURSE CONTENT

PART-A

1. Create a Scenario of Star, Ring and Bus topologies through simulation.
2. Simulate a stop and wait protocol and sliding window protocol
3. Create a Scenario to study the high-level data link control protocol (HDLC) through simulation
4. Simulate go back n protocol
5. Simulate Selective Repeat protocol

PART-B

1. Write a C program for error detection using CRC-CCITT (16-bits).
2. Write a C program to generate Hamming Code for error detection and correction.
3. Write a C program to implement Distance vector Routing algorithm
4. Write a C program to implement Open Shortest Path First Routing Algorithm
5. Write a C Program to implement Border Gateway Protocol (BGP)

Course Outcomes: On completion of this course, students are able to:

COs	**Course Outcomes** with ***Action verbs*** for the Course topics
CO1	Illustrate networking concepts using C programming language
CO2	Demonstrate networking concepts using NS2 simulator

Course Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2
1	3	2	2		1							2
2	3	2	2	2	2							2

CONTENT

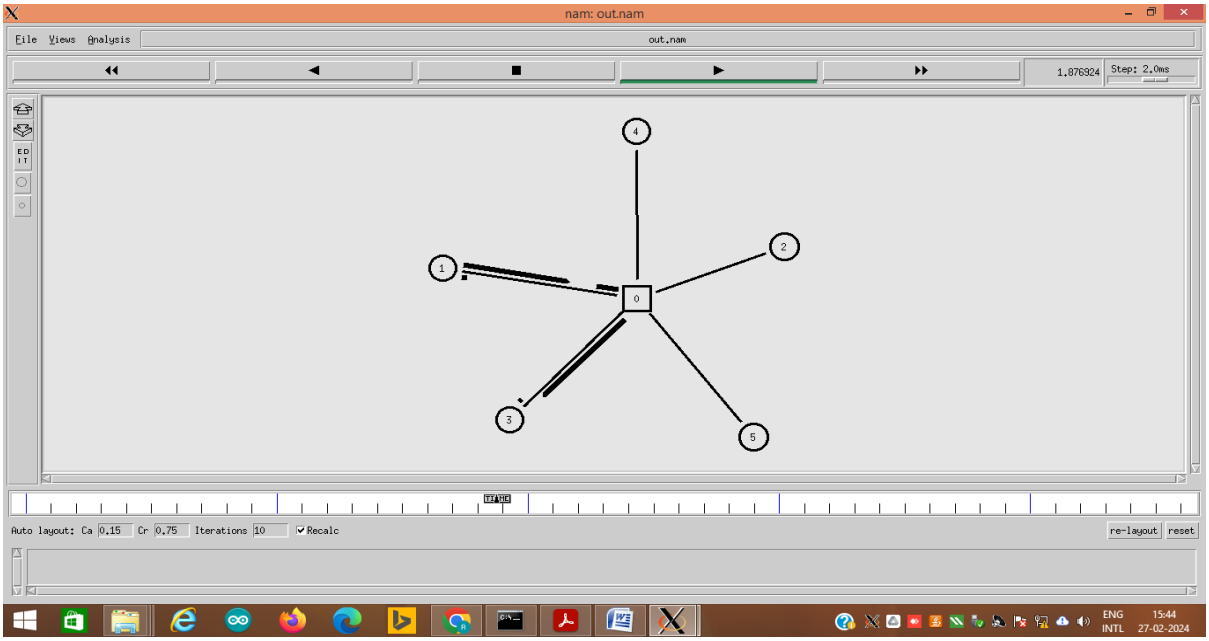
PART A

1. **Create a Scenario of Star, Ring and Bus topologies through simulation**.

Star Topology

```
#Create a simulator object
 set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam on the trace file
    exec nam out.nam &
    exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#Change the shape of center node in a star topology
$n0 shape square
#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
```

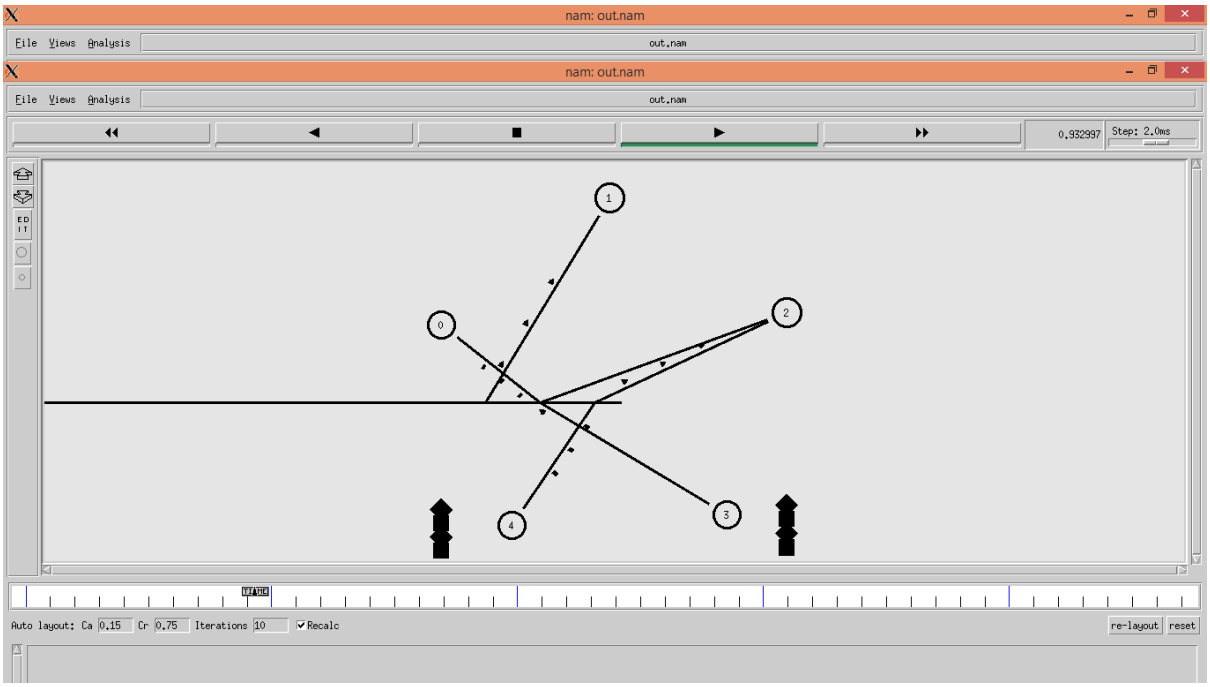
```
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```



Bus Topology

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
     #Close the trace file
     close $nf
     #Executenam on the trace file
      exec nam out.nam &
      exit 0
}
#Create five nodes
 set n0 [$ns node]
set n1 [$ns node]
```

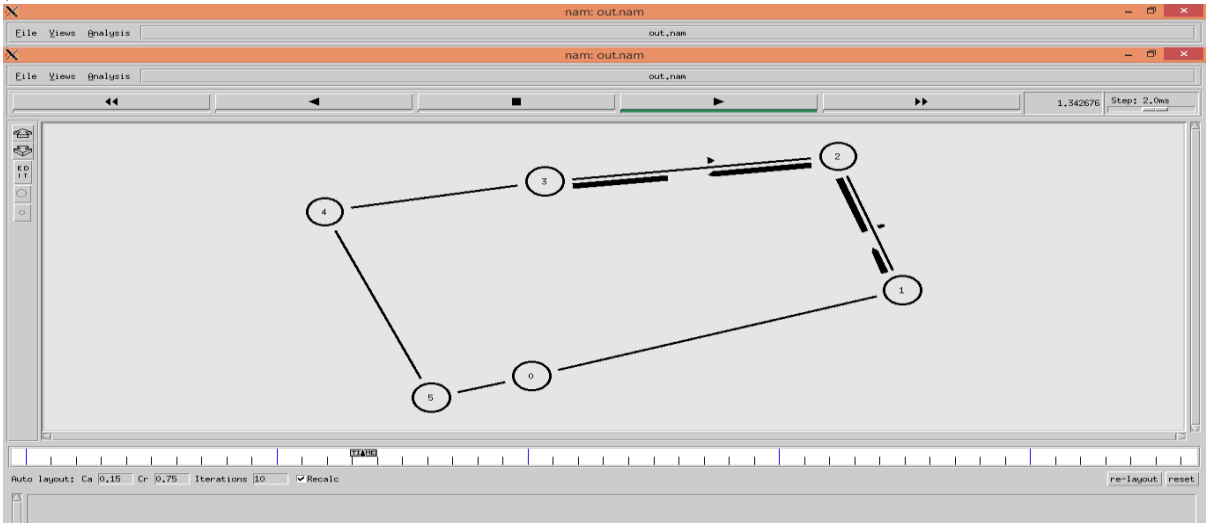
```
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Create Lan between the nodes
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail
MAC/Csma/Cd Channel]
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```



Ring Topology

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
     #Close the trace file
      close $nf
    #Executenam on the trace file
     exec nam out.nam &
    exit0
}
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
```

$ns run

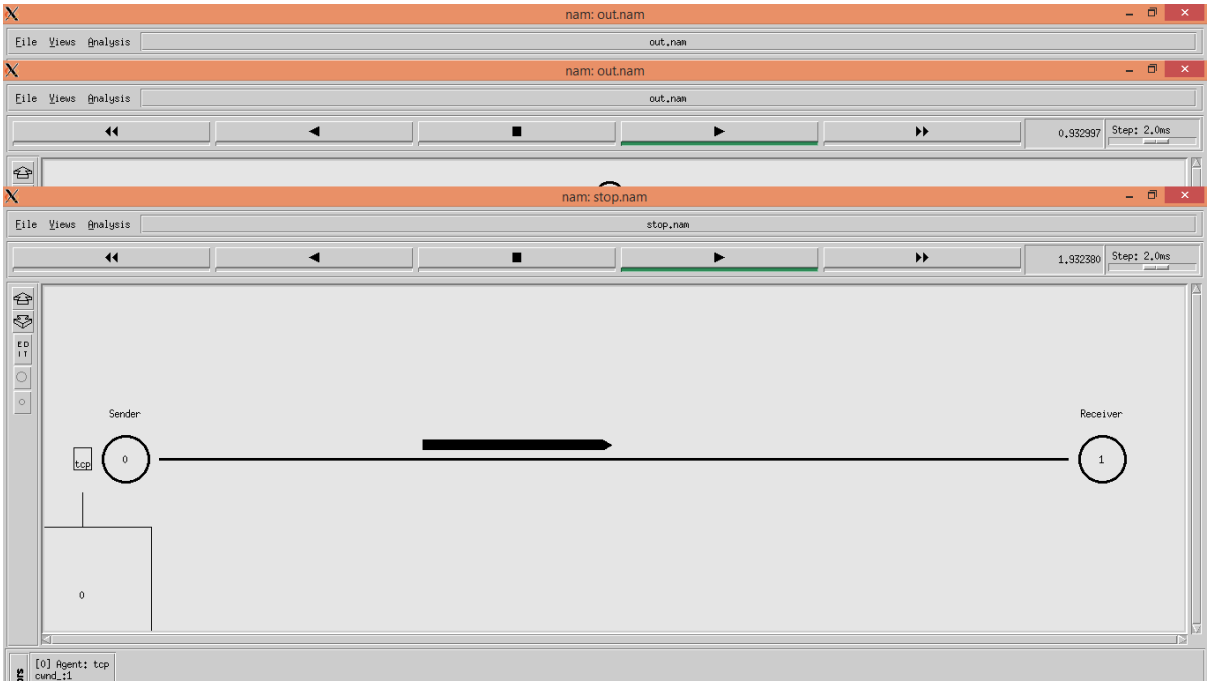


2. Simulate a stop and wait protocol and sliding window protocol

Stop and Wait protocol

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"
set nf [open stop.nam w]
$ns namtrace-all $nf
set f [open stop.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_ true
set tcp [new Agent/TCP]
$tcp set window_ 1
$tcp set maxcwnd_ 1
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
```

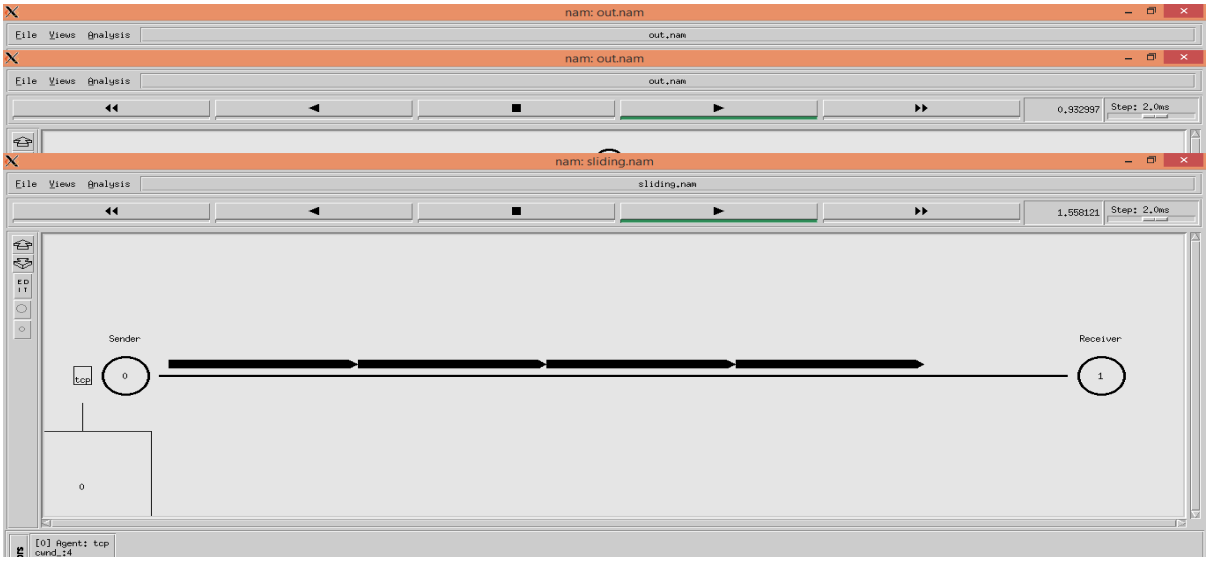
```
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0\""
$ns at 0.35 "$ns trace-annotate \"Receive Ack_0\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_1\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_1\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_2\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_2 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_3\""
$ns at 1.67 "$ns trace-annotate \"Receive Ack_3\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_4\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_4\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_5\""
$ns at 2.55 "$ns trace-annotate \"Receive Ack_5      \""
$ns at 2.75 "$ns trace-annotate \"Send Packet_6\""
$ns at 2.99 "$ns trace-annotate \"Receive Ack_6\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""
proc finish {} {
    global ns nf
    $ns flush-trace close $nf
     puts "running nam..."
     exec nam stop.nam &
     exit 0
}
$ns run
```



Sliding window protocol

```
# sliding window mechanism with some features
# such as labeling, annotation, nam-graph, and window size monitoring
 set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"
set nf [open sliding.nam w]
$ns namtrace-all $nf
set f [open sliding.tr w]
$ns trace-all $f

$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_ true
set tcp [new Agent/TCP]
$tcp set windowInit_ 4
$tcp set maxcwnd_ 4
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Sliding Window with window size 4 (normal operation)\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0,1,2,3\""
$ns at 0.34 "$ns trace-annotate \"Receive Ack_0,1,2,3\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_4,5,6,7\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_4,5,6,7\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_8,9,10,11\""

$ns at 1.23 "$ns trace-annotate \"Receive Ack_8,9,10,11 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_12,13,14,15\""
$ns at 1.67 "$ns trace-annotate \"Receive Ack_12,13,14,15\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_16,17,18,19\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_16,17,18,19\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_20,21,22,23\""
$ns at 2.56 "$ns trace-annotate \"Receive Ack_24,25,26,27\""
$ns at 2.76 "$ns trace-annotate \"Send Packet_28,29,30,31\""
$ns at 3.00 "$ns trace-annotate \"Receive Ack_28\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""
```

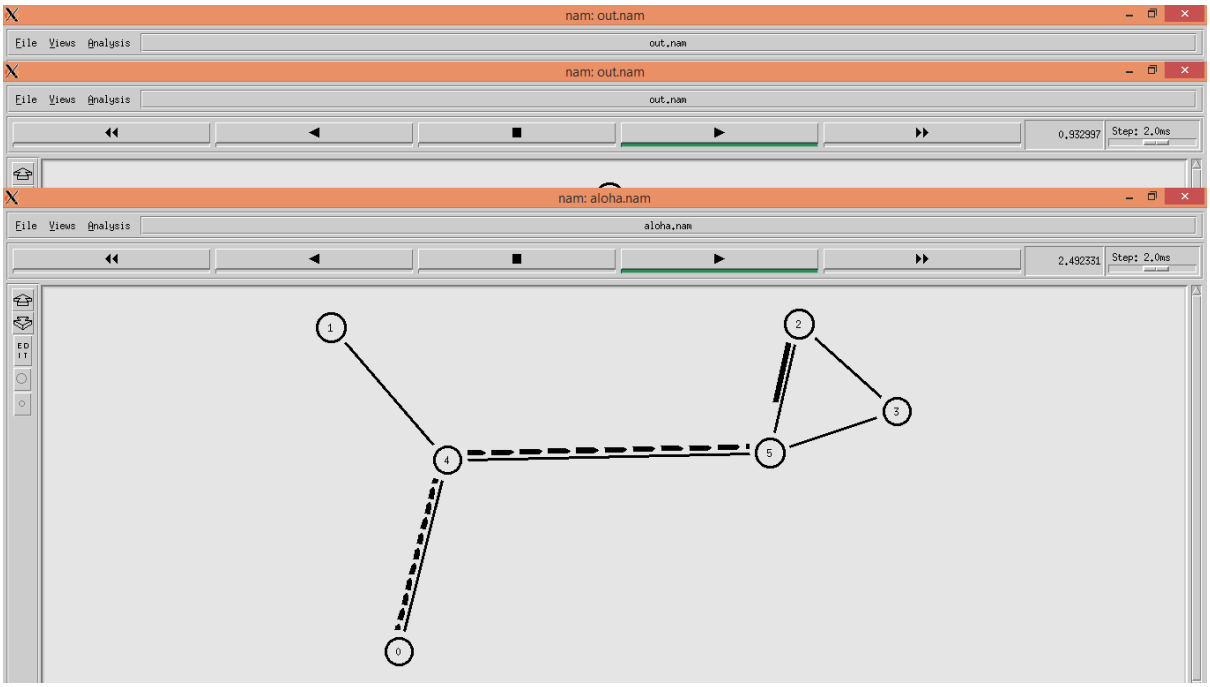
```
proc finish {} {
    global ns
    $ns flush-trace
    #close $nf
    puts "running nam..."
    exec nam sliding.nam &
   exit 0
}
$ns run
```



3. Create a Scenario to study the high-level data link control protocol (HDLC) through simulation

```
set ns [new Simulator]
#Tell the simulator to use dynamic routing
$ns rtproto DV
$ns macType Mac/Sat/UnslottedAloha
#Open the nam trace file
set nf [open aloha.nam w]
$ns namtrace-all $nf
#Open the output files
set f0 [open aloha.tr w]
$ns trace-all $f0
#Define a finish procedure
proc finish {} {
global ns f0 nf
$ns flush-trace
#Close the trace file
close $f0
close $nf
exec nam aloha.nam &
exit 0
}
```

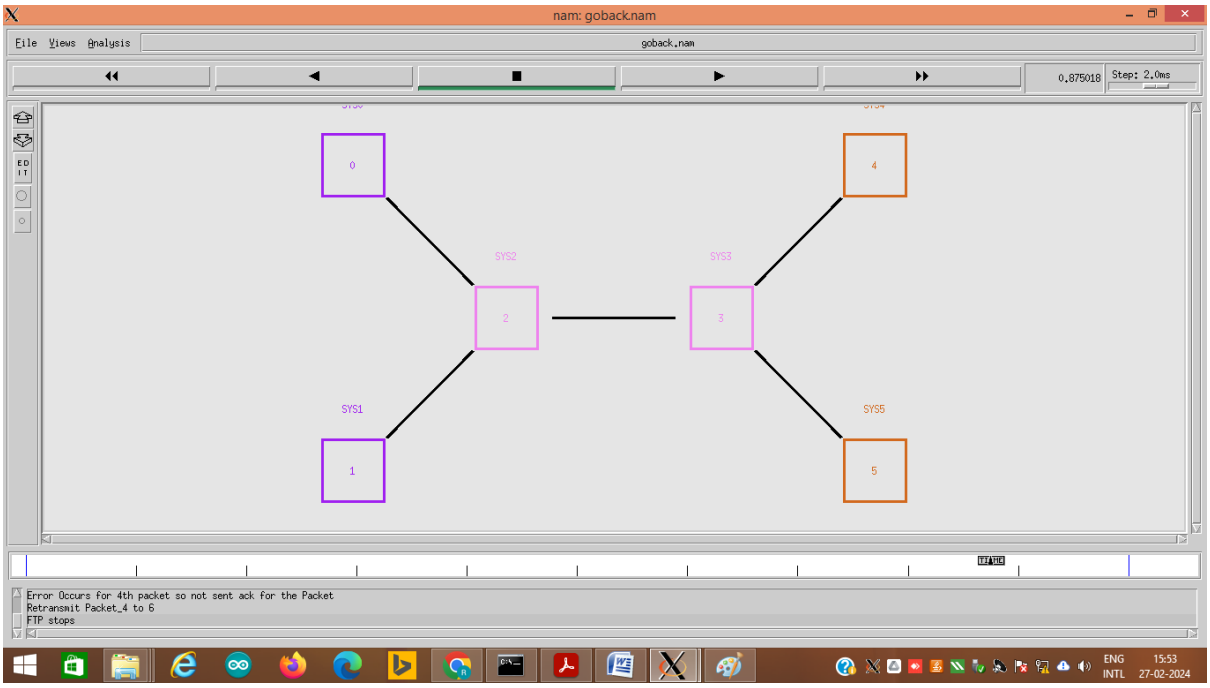
```
# Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
# Create duplex links between nodes with bandwidth and distance
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 1Mb 1ms DropTail
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link $n2 $n3 1Mb 50ms DropTail
# Create a duplex link between nodes 4 and 5 as queue position
$ns duplex-link-op $n4 $n5 queuePos 0.5
#Create a UDP agent and attach it to node n(0)
 set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n(2)
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n5 $n2
$ns rtmodel-at 2.0 up $n5 $n2
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```



4. Simulate go back n protocol

```
#send packets one by one
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 color "purple"
$n1 color "purple"
$n2 color "violet"
$n3 color "violet"
$n4 color "chocolate"
$n5 color "chocolate"
$n0 shape box ;
$n1 shape box ;
$n2 shape box ;
$n3 shape box ;
$n4 shape box ;
$n5 shape box ;
$ns at 0.0 "$n0 label SYS0"
$ns at 0.0 "$n1 label SYS1"
$ns at 0.0 "$n2 label SYS2"
$ns at 0.0 "$n3 label SYS3"
$ns at 0.0 "$n4 label SYS4"
$ns at 0.0 "$n5 label SYS5"
set nf [open goback.nam w]
$ns namtrace-all $nf
```

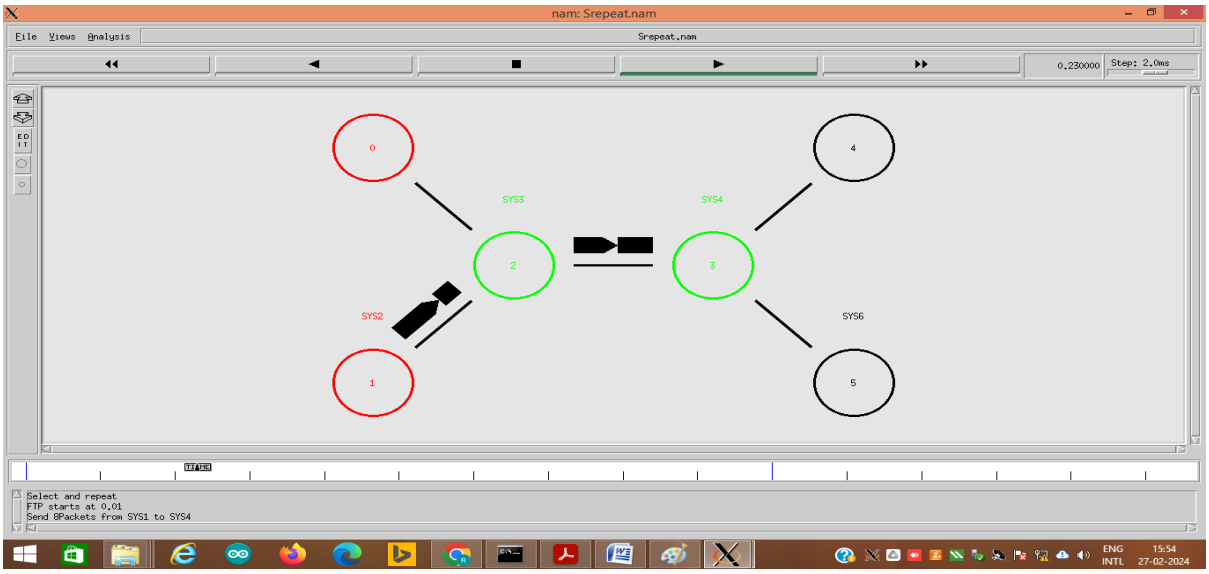
```
set f [open goback.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 20ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
$ns queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 20ms DropTail
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link $n2 $n3 1Mb 20ms DropTail
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 20ms DropTail
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link $n3 $n5 1Mb 20ms DropTail
$ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP]
$tcp set fid 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 6"
$ns at 0.06 "$tcp set maxcwnd 6"
$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.305 "$tcp set windowlnit 4"
$ns at 0.305 "$tcp set maxcwnd 4"
$ns at 0.368 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n4 $sink"
$ns at 1.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Goback N end\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 6Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs for 4th packet so not sent ack for the Packet\""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 to 6\""
$ns at 1.0 "$ns trace-annotate \"FTP stops\""
proc finish {} {
global ns nf
$ns flush-trace
close $nf
puts "filtering..."
#exec tclsh../bin/namfilter.tcl goback.nam
#puts "running nam..."
exec nam goback.nam &
exit 0
}
$ns run
```



5. Simulate Selective Repeat protocol

```
#send packets one by one
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 color "red"
$n1 color "red"
$n2 color "green"
$n3 color "green"
$n4 color "black"
$n5 color "black"
$n0 shape circle ;
$n1 shape circle ;
$n2 shape circle ;
$n3 shape circle ;
$n4 shape circle ;
$n5 shape circle ;
$ns at 0.0 "$n0 label SYS1"
$ns at 0.0 "$n1 label SYS2"
$ns at 0.0 "$n2 label SYS3"
$ns at 0.0 "$n3 label SYS4"
$ns at 0.0 "$n4 label SYS5"
$ns at 0.0 "$n5 label SYS6"
set nf [open Srepeat.nam w]
```

```
$ns namtrace-all $nf
set f [open Srepeat.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
$ns queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link $n3 $n5 1Mb 10ms DropTail
$ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP]
$tcp set fid 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 8"
$ns at 0.06 "$tcp set maxcwnd 8"
$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.30 "$tcp set windowlnit 1"
$ns at 0.30 "$tcp set maxcwnd 1"
$ns at 0.30 "$ns queue-limit $n3 $n4 10"
$ns at 0.47 "$ns detach-agent $n1 $tcp;$ns detach-agent $n4 $sink"
$ns at 1.75 "finish"
$ns at 0.0 "$ns trace-annotate \"Select and repeat\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 8Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs in 4th packet \""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 from SYS1 to SYS4\""
$ns at 1.5 "$ns trace-annotate \"FTP stops\""
proc finish {} {
global ns nf
$ns flush-trace
close $nf
puts "filtering..."
#exec tclsh../bin/namfilter.tcl srepeat.nam
#puts "running nam..."
exec nam Srepeat.nam &
exit 0
}
$ns run
```



PART-B

1. **Write a C program for error detection using CRC-CCITT (16-bits).**

```
// Include headers
#include<stdio.h>
#include<string.h>
// length of the generator polynomial
#define N strlen(gen_poly)
// data to be transmitted and received
char data[28];
// CRC value
char check_value[28];
// generator polynomial
char gen_poly[10];
// variables
int data_length,i,j;
// function that performs XOR operation
void XOR(){
   // if both bits are the same, the output is 0
   // if the bits are different the output is 1
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
```

```
}
// Function to check for errors on the receiver side
void receiver(){
// get the received data
    printf("Enter the received data: ");
    scanf("%s", data);
    printf("\n-----------------------------\n");
    printf("Data received: %s", data);
// Cyclic Redundancy Check
    crc();
// Check if the remainder is zero to find the error
    for(i=0;(i<N-1) && (check_value[i]!='1');i++);
        if(i<N-1)
          printf("\nError detected\n\n");
        else
            printf("\nNo error detected\n\n");
}

void crc(){
    // initializing check_value
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
    // check if the first bit is 1 and calls XOR function
        if(check_value[0]=='1')
            XOR();
// Move the bits by 1 position for the next computation
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        // appending a bit from data
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
// loop until the data ends
}

int main()
```

```
{
    // get the data to be transmitted
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    // get the generator polynomial
    scanf("%s",gen_poly);
    // find the length of data
    data_length=strlen(data);
    // appending n-1 zeros to the data
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n---------------------------------------");
// print the data with padded zeros
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n---------------------------------------");
// Cyclic Redundancy Check
    crc();
// print the computed check value
    printf("\nCRC or Check value is : %s",check_value);
// Append data with check_value(CRC)
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n---------------------------------------");
// printing the final data to be sent
    printf("\n Final data to be sent : %s",data);
    printf("\n---------------------------------------\n");
// Calling the receiver function to check errors
    receiver();
        return 0;
}
```

Output:

```
Enter data to be transmitted: 1001101
Enter the Generating polynomial: 1011
---------------------------------------
Data padded with n-1 zeros : 1001101000
---------------------------------------
CRC or Check value is : 101
---------------------------------------
Final data to be sent : 1001101101
---------------------------------------
Enter the received data: 1001101101
----------------------------
Data received: 1001101101
No error detected
```

2. Write a C program to generate Hamming Code for error detection and correction.

```
#include <stdio.h>
#include <math.h>
int input[32];
int code[32];
int ham_calc(int,int);
void main()
{
        int n,i,p_n = 0,c_l,j,k;
        printf("Please enter the length of the Data Word: ");
        scanf("%d",&n);
        printf("Please enter the Data Word:\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&input[i]);
        }

        i=0;
        while(n>(int)pow(2,i)-(i+1))
        {
                p_n++;
                i++;
        }

        c_l = p_n + n;

        j=k=0;
```

```
	for(i=0;i<c_l;i++)
	{

		if(i==((int)pow(2,k)-1))
		{
			code[i]=0;
			k++;
		}
		else
		{
			code[i]=input[j];
			j++;
		}
	}
	for(i=0;i<p_n;i++)
	{
		int position = (int)pow(2,i);
		int value = ham_calc(position,c_l);
		code[position-1]=value;
	}
	printf("\nThe calculated Code Word is: ");
	for(i=0;i<c_l;i++)
		printf("%d",code[i]);
	printf("\n");
	printf("Please enter the received Code Word:\n");
	for(i=0;i<c_l;i++)
		scanf("%d",&code[i]);

	int error_pos = 0;
	for(i=0;i<p_n;i++)
	{
		int position = (int)pow(2,i);
		int value = ham_calc(position,c_l);
		if(value != 0)
			error_pos+=position;
	}
	if(error_pos == 1)
		printf("The received Code Word is correct.\n");
	else
		printf("Error at bit position: %d\n",error_pos);
}
int ham_calc(int position,int c_l)
{
	int count=0,i,j;
	i=position-1;
	while(i<c_l)
	{
		for(j=i;j<i+position;j++)
		{
			if(code[j] == 1)
```

```
                                        count++;
                        }
                        i=i+2*position;
                }
                if(count%2 == 0)
                        return 0;
                else
                        return 1;
}
```

output:-

```
Please enter the length of the Data Word: 8
Please enter the Data Word:
1
1
0
1
0
0
1
1

The calculated Code Word is: 011110100011
Please enter the received Code Word:
0
1
1
1
1
1
1
0
0
0
1
1
Error at bit position: 6
```

3. Write a C program to implement Distance vector Routing algorithm

```
#include<stdio.h>
int dist[50][50],temp[50][50],n,i,j,k,x;
void dvr();
int main()
{
   printf("\nEnter the number of nodes : ");
```

```
   scanf("%d",&n);
   printf("\nEnter the distance matrix :\n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         scanf("%d",&dist[i][j]);
         dist[i][i]=0;
         temp[i][j]=j;
      }
      printf("\n");
            }
    dvr();
    printf("enter value of i &j:");
    scanf("%d",&i);
              scanf("%d",&j);
              printf("enter the new cost");
              scanf("%d",&x);
              dist[i][j]=x;
              printf("After update\n\n");
    dvr();
              return 0;
}
void dvr()
{
             for (i = 0; i < n; i++)
          for (j = 0; j < n; j++)
             for (k = 0; k < n; k++)
                          if (dist[i][k] + dist[k][j] < dist[i][j])
                          {
                          dist[i][j] = dist[i][k] + dist[k][j];
                          temp[i][j] = k;
                          }

             for(i=0;i<n;i++)
      {
         printf("\n\nState value for router %d is \n",i+1);
         for(j=0;j<n;j++)
            printf("\t\nnode %d via %d Distance%d",j+1,temp[i][j]+1,dist[i][j]);
      }
   printf("\n\n");

}
```

output:-

```
Enter the number of nodes : 4

Enter the distance matrix :
0
```

```
12
9
16

20
0
18
4

15
8
0
32

41
24
51
0



State value for router 1 is

node 1 via 1 Distance0
node 2 via 2 Distance12
node 3 via 3 Distance9
node 4 via 4 Distance16

State value for router 2 is

node 1 via 1 Distance20
node 2 via 2 Distance0
node 3 via 3 Distance18
node 4 via 4 Distance4

State value for router 3 is

node 1 via 1 Distance15
node 2 via 2 Distance8
node 3 via 3 Distance0
node 4 via 2 Distance12

State value for router 4 is

node 1 via 1 Distance41
node 2 via 2 Distance24
node 3 via 2 Distance42
node 4 via 4 Distance0

enter value of i &j:
```

```
1
3
enter the new cost
68
After update



State value for router 1 is

node 1 via 1 Distance0
node 2 via 2 Distance12
node 3 via 3 Distance9
node 4 via 4 Distance16

State value for router 2 is

node 1 via 1 Distance20
node 2 via 2 Distance0
node 3 via 3 Distance18
node 4 via 3 Distance30

State value for router 3 is

node 1 via 1 Distance15
node 2 via 2 Distance8
node 3 via 3 Distance0
node 4 via 2 Distance12

State value for router 4 is

node 1 via 1 Distance41
node 2 via 2 Distance24
node 3 via 2 Distance42
node 4 via 4 Distance0
```

4. Write a C program to implement Open Shortest Path First Routing Algorithm

```
 #include <stdio.h>
#include <string.h>
int main()
{
int count,src_router,i,j,k,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];
 printf("\n Enter the no of routers");
scanf("%d",&count);
```

```
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
 printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
}
}
 printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
flag[src_router]=1;
for(i=0;i<count;i++)
{
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
{
v=w;
min=dist[w];
}
}
flag[v]=1;
for(w=0;w<count;w++)
{
if(!flag[w])
if(min+cost_matrix[v][w]<dist[w])
{
dist[w]=min+cost_matrix[v][w];
last[w]=v;
}
}
}
for(i=0;i<count;i++)
{
 printf("\n%d==>%d:Path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
 printf("\n<--%d",last[w]);w=last[w];
}
```

```
 printf("\n Shortest path cost:%d",dist[i]);
}
}
```

Output

```
Enter the no of routers 2
Enter the cost matrix values:
0->0:3
0->1:4
1->0:5
1->1:6
Enter the source router: 1
1==>0:Path taken:0
Shortest path cost:5
1==>1:Path taken: 1
Shortest path cost:6
```

5. Write a C Program to implement Border Gateway Protocol (BGP)

```
#include <stdio.h>
#include<conio.h>
int main()
{
int n;
int i,j,k;
int a[10][10],b[10][10];
printf("\n Enter the number of nodes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("\n Enter the distance between the host %d - %d:",i+1,j+1);
scanf("%d",&a[i][j]);
}}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
for(k=0;k<n;k++)
{
for(i=0;i<n;i++)
{
```

```
for(j=0;j<n;j++)
{
if(a[i][j]>a[i][k]+a[k][j])
{
a[i][j]=a[i][k]+a[k][j];
}}}}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
b[i][j]=a[i][j];
if(i==j)
{
b[i][j]=0;
}
}}
printf("\n The output matrix:\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
getch();
}
```

Output

Enter the number of nodes:4

Enter the distance between the host 1 - 1:5
Enter the distance between the host 1 - 2:9
Enter the distance between the host 1 - 3:6
Enter the distance between the host 1 - 4:4
Enter the distance between the host 2 - 1:2
Enter the distance between the host 2 - 2:1
Enter the distance between the host 2 - 3:8
Enter the distance between the host 2 - 4:3
Enter the distance between the host 3 - 1:6
Enter the distance between the host 3 - 2:1
Enter the distance between the host 3 - 3:4
Enter the distance between the host 3 - 4:2
Enter the distance between the host 4 - 1:5
Enter the distance between the host 4 - 2:1
Enter the distance between the host 4 - 3:8
Enter the distance between the host 4 - 4:2

5 9 6 4

2 1 8 3
6 1 4 2
5 1 8 2

The output matrix:
0 5 6 4
2 0 8 3
3 1 0 2
3 1 8 0