PROGRAM OUTCOMES (PO)

PO 1: Engineering knowledge

Apply the Mathematical knowledge and the basics of Science and Engineering to solve the problems pertaining to Electronics and Instrumentation Engineering.

PO2: Problem analysis

Identify and formulate Electrical and Electronics Engineering problems from research literature and be able to analyze the problem using first principles of Mathematics and Engineering Sciences

PO3: Design/development of solutions

Come out with solutions for the complex problems and to design system components or process that fulfil the particular needs taking into account public health and safety and the social, cultural and environmental issues.

PO4: Conduct investigations of complex problems

Draw well-founded conclusions applying the knowledge acquired from research and research methods including design of experiments, analysis and interpretation of data and synthesis of information and to arrive at significant conclusion.

PO5: Modern tool usage

Form, select and apply relevant techniques, resources and Engineering and IT tools for Engineering activities like electronic prototyping, modelling and control of systems and also being conscious of the limitations.

PO6: The engineer and society

Understand the role and responsibility of the Professional Electrical and Electronics Engineer and to assess societal, health, safety issues based on the reasoning received from the contextual knowledge.

PO7: Environment and sustainability

Be aware of the impact of professional Engineering solutions in societal and environmental contexts and exhibit the knowledge and the need for sustainable Development.

PO8: Ethics

Apply the principles of Professional Ethics to adhere to the norms of the engineering practice and to discharge ethical responsibilities.

PO9: Individual and team work

Function actively and efficiently as an individual or a member/leader of different teams and multidisciplinary projects.

PO10: Communication

Communicate efficiently the engineering facts with a wide range of engineering community and others, to understand and prepare reports and design documents; to make effective presentations and to frame and follow instructions

PO11: Project management and finance

Demonstrate the acquisition of the body of engineering knowledge and insight and Management Principles and to apply them as member / leader in teams and multidisciplinary environments.

PO12: Life-long learning

Recognize the need for self and life-long learning, keeping pace with technological challenges in the broadest sense.

EC3401 NETWORKS AND SECURITY

L	T	P	C
0	0	2	1

OBJECTIVES:

The student should be made to:

- Explain the Network Models, layers and functions.
- Categorize and classify the routing protocols.
- List the functions of the transport and application layer.
- Evaluate and choose the network security mechanisms.
- Discuss the hardware security attacks and countermeasures.

LIST OF EXPERIMENTS:

1. Implement the Data Link Layer framing methods,
i) Bit stuffing, (ii) Character stuffing
2. Implementation of Error Detection / Correction Techniques
i) LRC, (ii) CRC, (iii) Hamming code
3. Implementation of Stop and Wait, and Sliding Window Protocols
4. Implementation of Go back-N and Selective Repeat Protocols.
5. Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford).
6. Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes(Dijkstra's).
7. Data encryption and decryption using Data Encryption Standard algorithm.
8. Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm.
9. Implement Client Server model using FTP protocol.

CONTENT BEYOND SYLLABUS:

10. Simulate a Mobile Adhoc Network.
11. Implement Transport Control Protocol in Sensor Network

TOTAL: 30 PERIODS

OUTCOMES:

At the end of the course, the student should be able to

- Explain the Network Models, layers and functions.
- Categorize and classify the routing protocols.
- List the functions of the transport and application layer.
- Evaluate and choose the network security mechanisms.
- Discuss the hardware security attacks and countermeasures.

INDEX

List of Experiments Mapping with Cos, POs, and PSOs

S.No	Name of the Experiment	CO_S	PO_S	PSO_S	Page No
1	Implementation of Bit stuffing and Character stuffing	CO1	PO1,PO2,PO4,PO12	PSO1	1
2	Implementation of Error Detection / Correction Techniques	CO1	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	5
3	Implementation of Stop and Wait, and Sliding Window Protocols	CO1	PO1,PO2,PO3,PO4,PO5,PO12	PSO1, PSO3	13
4	Implementation of Go back-N and Selective Repeat Protocols	CO1	PO1,PO2,PO3,PO4,PO5,PO12	PSO1, PSO3	17
5	Implementation of Distance Vector Routing algorithm	CO2	PO1,PO2,PO3,PO4,PO12	PSO1	21
6	Implementation of Link State Routing algorithm	CO2	PO1,PO2,PO3,PO4,PO12	PSO1	23
7	Data encryption and decryption using Data Encryption Standard algorithm	CO4	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	27
8	Data encryption and decryption using RSA algorithm	CO4	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	45
9	Implement Client Server model using FTP protocol.	CO3	PO1,PO2,PO4,PO12	PSO1	47
10	Implement and realize the Network Topology - Star, Bus and Ring using NS2	CO3	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	53
11	Implement and perform the operation of CSMA/CD and CSMA/CA using NS2	CO3	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	59
CBS1	Simulate a Mobile Adhoc Network.	CO5	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	63
CBS2	Implement Transport Control Protocol in Sensor Network	CO5	PO1,PO2,PO3,PO4,PO5,PO12	PSO1	65

Justification of the mapping: (Experiments with POs)

Ex:No1	It involves strong theory in mathematics and engineering to understand the Data Link Layer framing methods (PO1). This course outcome provides strong idea to identify and analyze the problems on Data Link Layer framing methods (PO2). It instils the fair knowledge to develop solutions for complex problems on Data Link Layer framing methods (PO3). It provides the fair knowledge to find the solutions for complex problems on Communication between two desktop computers by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the Data Link Layer framing methods (PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No2	It involves strong theory in mathematics and engineering to understand the Error Detection / Correction Techniques (PO1). This course outcome provides strong idea to identify and analyze the problems on Error Detection / Correction Techniques (PO2). It instils the fair knowledge to develop solutions for complex problems on Error Detection / Correction Techniques (PO3). It provides the fair knowledge to find the solutions for complex problems on Error Detection / Correction Techniques by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the Error Detection / Correction Techniques (PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No3	It involves strong theory in mathematics and engineering to analyze the different protocols(PO1).This course outcome provides strong idea to identify and analyze the problems on the different protocols(PO2). It instils the fair knowledge to develop solutions for complex problems on the different protocols(PO3). It provides the fair knowledge to find the solutions for complex problems on the different protocols by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to analyze the different protocols(PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No4	It involves strong theory in mathematics and engineering to analyze the different protocols(PO1).This course outcome provides strong idea to identify and analyze the problems on the different protocols(PO2). It instils the fair knowledge to develop solutions for complex problems on the different protocols(PO3). It provides the fair knowledge to find the solutions for complex problems on the different protocols by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to analyze the different protocols(PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No5	This gives the strong knowledge on engineering and mathematical background to implement and compare the various routing algorithms(PO1). It provides fair idea to identify and analyze the problems on various routing algorithms with the fundamental knowledge in mathematics and engineering(PO2). It gives the fair knowledge to develop the solutions for complex problems on various routing algorithms(PO4). This provides the basic knowledge to develop the solution for problems on various routing algorithms by conducting experiments(PO4). It instills the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No6	This gives the strong knowledge on engineering and mathematical background to implement and compare the various routing algorithms(PO1). It provides fair idea to identify and analyze the problems on various routing algorithms with the fundamental knowledge in mathematics and engineering(PO2). It gives the fair knowledge to develop the solutions for complex problems on various routing algorithms(PO4). This provides the basic knowledge to develop the solution for problems on various routing algorithms by conducting experiments(PO4). It instills the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).

Ex:No7	It involves strong theory in mathematics and engineering to understand the encryption standards(PO1). This course outcome provides strong idea to identify and analyze the problems on encryption standards (PO2). It instils the fair knowledge to develop solutions for complex problems on encryption standards (PO3). It provides the fair knowledge to find the solutions for complex problems on encryption standards by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the encryption standards (PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No8	It involves strong theory in mathematics and engineering to understand the encryption standards(PO1). This course outcome provides strong idea to identify and analyze the problems on encryption standards (PO2). It instils the fair knowledge to develop solutions for complex problems on encryption standards (PO3). It provides the fair knowledge to find the solutions for complex problems on encryption standards by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the encryption standards (PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No9	It involves strong theory in mathematics and engineering to understand the Communication between two desktop computers(PO1). This course outcome provides strong idea to identify and analyze the problems on Communication between two desktop computers(PO2). It instils the fair knowledge to develop solutions for complex problems on Communication between two desktop computers(PO3). It provides the fair knowledge to find the solutions for complex problems on Communication between two desktop computers by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the Communication between two desktop computers(PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No10	It involves strong theory in mathematics and engineering to understand the Communication between two desktop computers(PO1). This course outcome provides strong idea to identify and analyze the problems on Communication between two desktop computers(PO2). It instils the fair knowledge to develop solutions for complex problems on Communication between two desktop computers(PO3). It provides the fair knowledge to find the solutions for complex problems on Communication between two desktop computers by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the Communication between two desktop computers(PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
Ex:No11	It involves strong theory in mathematics and engineering to understand the Communication between two desktop computers(PO1). This course outcome provides strong idea to identify and analyze the problems on Communication between two desktop computers(PO2). It instils the fair knowledge to develop solutions for complex problems on Communication between two desktop computers(PO3). It provides the fair knowledge to find the solutions for complex problems on Communication between two desktop computers by conducting experiments(PO4). It gives the basic knowledge about the simulation tools to understand the Communication between two desktop computers(PO5). It instils the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).
CBS 1	It involves strong theory in mathematics and engineering to apply the simulation tool(PO1). This course outcome provides fair idea to identify and analyze the problems to apply the simulation tool(PO2). It instils the basic knowledge to develop solutions for complex problems to apply the simulation tool(PO3). It provides the basic knowledge to find the solutions for complex problems to apply the simulation tool by conducting experiments(PO4). It gives the fair idea about the usage of simulation tools(PO5). It instills the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).

CBS2	It involves strong theory in mathematics and engineering to apply the simulation tool(PO1). This course outcome provides fair idea to identify and analyze the problems to apply the simulation tool(PO2). It instils the basic knowledge to develop solutions for complex problems to apply the simulation tool(PO3). It provides the basic knowledge to find the solutions for complex problems to apply the simulation tool by conducting experiments(PO4). It gives the fair idea about the usage of simulation tools(PO5). It instills the basic idea to engage in independent and life-long learning in the broadest context of technological change(PO12).

EX.NO.1 IMPLEMENTATION OF BIT STUFFING AND CHARACTER STUFFING

AIM:

To write a C program to implement bit stuffing and character stuffing.

Software Required:

Turbo C, C++

ALGORITHM:

1. Start the program.
2. Enter the frame size.
3. Enter the data frame.
4. Check the stuffed bit once the frame sequence encountered 5 consecutive 1's.
5. Stop the program

PROGRAM (BIT STUFFING):

```
#include<stdio.h>
#include<string.h>
int main()
{
   int a[20],b[30],i,j,k,count,n;
   printf("Enter frame size (Example: 8):");
   scanf("%d",&n);
   printf("Enter the frame in the form of 0 and 1 :");
   for(i=0; i<n; i++)
      scanf("%d",&a[i]);
   i=0;
   count=1;
   j=0;
   while(i<n)
   {
      if(a[i]==1)
```

```
        {
            b[j]=a[i];
            for(k=i+1; a[k]==1 && k<n && count<5; k++)
            {
                j++;
                b[j]=a[k];
                count++;
                if(count==5)
                {
                    j++;
                    b[j]=0;
                }
                i=k;
            }
        }
        else
        {
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing :");
    for(i=0; i<j; i++)
        printf("%d",b[i]);
    return 0;
}
```

PROGRAM (CHARECTER STUFFING):

```
#include<stdio.h>
#include<string.h>
main()
{
    char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3];
    int i, j, p = 0, q = 0;
    clrscr();
    printf("Enter characters to be stuffed:");
    scanf("%s", a);
    printf("\nEnter a character that represents starting delimiter:");
    scanf(" %c", &sd);
    printf("\nEnter a character that represents ending delimiter:");
    scanf(" %c", &ed);
    x[0] = s[0] = s[1] = sd;
    x[1] = s[2] = '\0';
    y[0] = d[0] = d[1] = ed;
    d[2] = y[1] = '\0';
    strcat(fs, x);
    for(i = 0; i < strlen(a); i++)
    {
       t[0] = a[i];
       t[1] = '\0';
       if(t[0] == sd)
          strcat(fs, s);
       else if(t[0] == ed)
          strcat(fs, d);
       else
```

```
        strcat(fs, t);
    }
    strcat(fs, y);
    printf("\n After stuffing:%s", fs);
    getch();
}
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT:

EX.NO.2 IMPLEMENTATION OF ERROR DETECTION AND ERROR CORRECTION TECHNIQUES (LRC, CRC, HAMMING CODE)

AIM:

To write a C program to implement the error detection and error correction techniques.

Software Required:

Turbo C, C++

ALGORITHM:

1. Start the program.
2. Transmit the data bit by bit.
3. Check for the error.
4. If there is error, correct the error.
5. Stop the program

PROGRAM (LRC):

```
void main()

{

int l1,bit[100],count=0,i,choice;

clrscr();

printf("Enter the length of data stream: ");

scanf("%d",&l1);

printf("\nEnter the data stream ");

for(i=0;i {

scanf("%d",&bit[i]);

if(bit[i]==1)

count=count+1;

}

printf("Number of 1's are %d",count);

printf("\nEnter the choice to implement parity bit");

printf("\n1-Sender side\n2-Receiver side\n");

scanf("%d",&choice);
```

```
switch(choice)
{
case 1:
if(count%2==0)
bit[l1]=0;
else
bit[l1]=1;

printf("\nThe data stream after adding parity bit is\n");
for(i=0;i<=l1;i++)
printf("%d",bit[i]);
break;

case 2:
if(count%2==0)
printf("There is no error in the received data stream");
else
printf("There is error in the received data stream");
break;

default:
printf("Invalid choice");
break;
}
getch();
}
```

PROGRAM (CRC):

```
#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;
}
```

PROGRAM (HAMMING CODE):

```
#include<stdio.h>
#include<conio.h>
void main() {
        int data[7],rec[7],i,c1,c2,c3,c;
        printf("this works for message of 4bits in size \nenter message bit one by one:  ");
        scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);
        data[6]=data[0]^data[2]^data[4];
        data[5]=data[0]^data[1]^data[4];
        data[3]=data[0]^data[1]^data[2];
        printf("\nthe encoded bits are given below: \n");
        for (i=0;i<7;i++) {
                printf("%d ",data[i]);
        }
        printf("\nenter the received data bits one by one: ");
        for (i=0;i<7;i++) {
                scanf("%d",&rec[i]);
        }
        c1=rec[6]^rec[4]^rec[2]^rec[0];
        c2=rec[5]^rec[4]^rec[1]^rec[0];
        c3=rec[3]^rec[2]^rec[1]^rec[0];
        c=c3*4+c2*2+c1 ;
        if(c==0) {
                printf("\nThere is no error: ");
```

```
        } else {
                printf("\nerron on the postion: %d\nthe correct message is \n",c);
                if(rec[7-c]==0)
                                        rec[7-c]=1; else
                                        rec[7-c]=0;
                for (i=0;i<7;i++) {
                        printf("%d ",rec[i]);
                }
        }
        getch();
}
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT:

EX .NO: 3a IMPLEMENTATION OF STOP AND WAIT PROTOCOL

AIM:

To write a c program to perform stop and wait protocol.

Software Required:

Turbo C, C++

ALOGORITHM:

1. Start the program
2. Get the frame size from user
3. To create frame based on user request
4. To send frames to server to client side
5. Stop the program

Program:

```
#include<stdio.h>
 void main()
 {
 int i,f;
 clrscr();
 printf("enter no. of frame");
 scanf("%d", &f);
 for(i=1;i<=f;i++)
 {
 printf("\n sender");
 printf("\n send frame %d",i);
 if(i<=f)
 {
 printf("\n receive frame %d",i);
 printf("\n ACK frame %d",i);
 }
```

```
}
getch();
}
```

Result:

Thus the program of Stop and Wait protocol has been executed successfully and the output was verified.

EX. NO: 3b IMPLEMENTATION OF SLIDING WINDOW PROTOCOL

AIM:

To write a 'C' program for sliding window protocol

Software Required:

Turbo C, C++

ALGORITHM:

1. Start the program
2. Get the frame size from user
3. To create frame based on user request
4. To send frame to server from client side
5. If your frame each server it will send ACK signal to client
6. Stop the program

Program:

```
#include<stdio.h>
void main()
{
int i,f,w,frame[50];
clrscr();
printf("enter the frame size");
scanf("%d",&f);
printf("enter the window size");
scanf("%d",&w);
printf("enter the frame:");
for(i=1;i<=f;i++)
{
scanf("%d",&frame[i]);
}
for(i=1;i<=f;i++)
```

```
{
if((i%w)==0)
{
printf("%d",frame[i]);
printf("the frame received \n");
}
else
{
printf("%d",frame[i]);
}
}
if((i%w)!=0)
{
printf("\n the frame received\n");
}
getch();
}
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

Result:

EX.NO:4 IMPLEMENTATION AND STUDY OF GO BACK-N AND SELECTIVE REPEAT PROTOCOL

AIM:

To write the C program to perform Go back-N and selective repeat protocol.

Software Required:

Turbo C, C++

ALGRORITHM:

1. Start the program.
2. Get the frame size from the user.
3. To create the frame based on the request.
4. To send frames to server from client.
5. If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
6. Stop the program.

PROGRAM:

Go back N-protocol;

```
#include<stdio.h>
void main()
{
int i, windowsize,sent=0,ack;
printf("enter windowsize\n");
scanf("%d",&windowsize);

while(1)
{
for(i=0;i<windowsize;i++)
{
```

```
printf("Frame %d has been transmitted\n",sent);
sent++;
if(sent==windowsize)
break;
}
printf("\n please enter the last acknowledgement receiver\n");
scanf("%d",&ack);
if(ack==windowsize)
break;
else
sent=ack;
}
getch();
}
```

SELECTIVE REPEAT PROTOCOL:

```
#include<stdio.h>
void main()
{
int windowsize,sent=0,nak,i;
clrscr();
printf("enter the window size");
scanf("%d",&windowsize);
for(i=0;i<windowsize;i++)
{
printf("frame %d has been transmitted \n", sent);
sent++;
```

```
if(sent==windowsize)
break;
}
printf("\n please enter the negative acknowledgement recieved");
scanf("%d",&nak);
if(nak==windowsize)
{
printf("successful");
}
else
printf("frame %d has been retransmitted ",nak);
getch();

}
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT:

EX.NO:5 IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM

Aim:

To write a ns2 program for implementing distance vector routing algorithm.

Software Required:

Turbo C, C++

Algorithm:

1: Start the program.
2: Write the program
3: Enter the number of nodes.
4: Enter the cost matrix..
5: Stop the program.

Program:

```
#include<stdio.h>
struct node
{
        unsigned dist[20];
        unsigned from[20];
}rt[10];
int main()
{
        int dmat[20][20];
        int n,i,j,k,count=0;
        printf("\nEnter the number of nodes : ");
        scanf("%d",&n);
        printf("\nEnter the cost matrix :\n");
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                {
                        scanf("%d",&dmat[i][j]);
                        dmat[i][i]=0;
                        rt[i].dist[j]=dmat[i][j];
                        rt[i].from[j]=j;
                }
                do
                {
                        count=0;
```

```
                        for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        for(k=0;k<n;k++)
                                if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
                                {
                                        rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                                        rt[i].from[j]=k;
                                        count++;
                                }
                }while(count!=0);
                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,rt[i].from[j]+1,rt[i].dist[j]);
                        }
                }
        printf("\n\n");
}
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

Result:

EX.NO:6 IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM

Aim:

To write a ns2 program for implementing link state routing algorithm.

Software Required:

Turbo C, C++

Algorithm:

1: Start the program.
2: Write the program
3: Enter the number of nodes.
4: Enter the cost matrix..
5: Stop the program.

Program:

```
#include<stdio.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]);

}

}
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

Result:

EX.NO:7 ENCRYPTION AND DECRYPTION USING DES ALGORITHM

AIM:

To write a c program to perform encryption and decryption using DES algorithm.

SOFTWARE REQUIRED:

Turbo C, C++

ALGORITHM:

1. Start the program.
2. Transmit the data list by bit.
3. Check for the error.
4. If there is error, correct the error.
5. Stop the program

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <time.h>

int IP[] =
{
   58, 50, 42, 34, 26, 18, 10, 2,
   60, 52, 44, 36, 28, 20, 12, 4,
   62, 54, 46, 38, 30, 22, 14, 6,
   64, 56, 48, 40, 32, 24, 16, 8,
   57, 49, 41, 33, 25, 17,  9, 1,
   59, 51, 43, 35, 27, 19, 11, 3,
   61, 53, 45, 37, 29, 21, 13, 5,
   63, 55, 47, 39, 31, 23, 15, 7
};

int E[] =
{
   32,  1,  2,  3,  4,  5,
   4,  5,  6,  7,  8,  9,
```

```
    8,  9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32,  1
};

int P[] =
{
    16,  7, 20, 21,
    29, 12, 28, 17,
    1, 15, 23, 26,
    5, 18, 31, 10,
    2,  8, 24, 14,
    32, 27,  3,  9,
    19, 13, 30,  6,
    22, 11,  4, 25
};

int FP[] =
{
    40, 8, 48, 16, 56, 24, 64, 32,
    39, 7, 47, 15, 55, 23, 63, 31,
    38, 6, 46, 14, 54, 22, 62, 30,
    37, 5, 45, 13, 53, 21, 61, 29,
    36, 4, 44, 12, 52, 20, 60, 28,
    35, 3, 43, 11, 51, 19, 59, 27,
    34, 2, 42, 10, 50, 18, 58, 26,
    33, 1, 41,  9, 49, 17, 57, 25
};

int S1[4][16] =
{
        14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7,
        0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8,
        4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0,
        15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13
};
```

```
int S2[4][16] =
{
   15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10,
   3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5,
   0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15,
   13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9
};

int S3[4][16] =
{
   10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8,
   13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1,
   13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7,
   1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12
};

int S4[4][16] =
{
   7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15,
   13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9,
   10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4,
   3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14
};

int S5[4][16] =
{
   2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9,
   14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6,
   4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14,
   11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3
};

int S6[4][16] =
{
   12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11,
   10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8,
   9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6,
   4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13
};
```

```
int S7[4][16]=
{
   4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1,
   13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6,
   1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2,
   6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12
};

int S8[4][16]=
{
   13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7,
   1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2,
   7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8,
   2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11
};

int PC1[] =
{
   57, 49, 41, 33, 25, 17,  9,
   1, 58, 50, 42, 34, 26, 18,
   10,  2, 59, 51, 43, 35, 27,
   19, 11,  3, 60, 52, 44, 36,
   63, 55, 47, 39, 31, 23, 15,
   7, 62, 54, 46, 38, 30, 22,
   14,  6, 61, 53, 45, 37, 29,
   21, 13,  5, 28, 20, 12,  4
};

int PC2[] =
{
   14, 17, 11, 24,  1,  5,
   3, 28, 15,  6, 21, 10,
   23, 19, 12,  4, 26,  8,
   16,  7, 27, 20, 13,  2,
   41, 52, 31, 37, 47, 55,
   30, 40, 51, 45, 33, 48,
   44, 49, 39, 56, 34, 53,
   46, 42, 50, 36, 29, 32
};
```

```
int SHIFTS[] = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };

FILE* out;
int LEFT[17][32], RIGHT[17][32];
int IPtext[64];
int EXPtext[48];
int XORtext[48];
int X[8][6];
int X2[32];
int R[32];
int key56bit[56];
int key48bit[17][48];
int CIPHER[64];
int ENCRYPTED[64];

void expansion_function(int pos, int text)
{
   for (int i = 0; i < 48; i++)
   {
      if (E[i] == pos + 1) {
         EXPtext[i] = text;
      }
   }
}

int initialPermutation(int pos, int text)
{
   int i;
   for (i = 0; i < 64; i++)
   {
      if (IP[i] == pos + 1) {
         break;
      }
   }
   IPtext[i] = text;
}

int F1(int i)
{
   int r, c, b[6];
```

```
    for (int j = 0; j < 6; j++) {
        b[j] = X[i][j];
    }

    r = b[0] * 2 + b[5];
    c = 8 * b[1] + 4 * b[2] + 2 * b[3] + b[4];

    if (i == 0) {
        return S1[r][c];
    }
    else if (i == 1) {
        return S2[r][c];
    }
    else if (i == 2) {
        return S3[r][c];
    }
    else if (i == 3) {
        return S4[r][c];
    }
    else if (i == 4) {
        return S5[r][c];
    }
    else if (i == 5) {
        return S6[r][c];
    }
    else if (i == 6) {
        return S7[r][c];
    }
    else if (i == 7) {
        return S8[r][c];
    }
}

int XOR(int a, int b) {
    return (a ^ b);
}

int ToBits(int value)
{
```

```
    int k, j, m;
    static int i;

    if (i % 32 == 0) {
        i = 0;
    }

    for (j = 3; j >= 0; j--)
    {
        m = 1 << j;
        k = value & m;
        if (k == 0) {
            X2[3 - j + i] = '0' – 48;
        }
        else {
            X2[3 - j + i] = '1' – 48;
        }
    }

    i = i + 4;
}

int SBox(int XORtext[])
{
    int k = 0;
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 6; j++) {
            X[i][j] = XORtext[k++];
        }
    }

    int value;
    for (int i = 0; i < 8; i++)
    {
        value = F1(i);
        ToBits(value);
    }
}
```

```
int PBox(int pos, int text)
{
   int i;
   for (i = 0; i < 32; i++)
   {
      if (P[i] == pos + 1) {
         break;
      }
   }
   R[i] = text;
}

void cipher(int Round, int mode)
{
   for (int i = 0; i < 32; i++) {
      expansion_function(i, RIGHT[Round – 1][i]);
   }

   for (int i = 0; i < 48; i++)
   {
      if (mode == 0) {
         XORtext[i] = XOR(EXPtext[i], key48bit[Round][i]);
      }
      else {
         XORtext[i] = XOR(EXPtext[i], key48bit[17 – Round][i]);
      }
   }

   SBox(XORtext);

   for (int i = 0; i < 32; i++) {
      PBox(i, X2[i]);
   }

   for (int i = 0; i < 32; i++) {
      RIGHT[Round][i] = XOR(LEFT[Round – 1][i], R[i]);
   }
}

void finalPermutation(int pos, int text)
```

```
{
   int i;
   for (i = 0; i < 64; i++)
   {
      if (FP[i] == pos + 1) {
         break;
      }
   }
   ENCRYPTED[i] = text;
}

void convertToBinary(int n)
{
   int k, m;
   for (int i = 7; i >= 0; i--)
   {
      m = 1 << i;
      k = n & m;

      if (k == 0) {
         fprintf(out, "0");
      }
      else {
         fprintf(out, "1");
      }
   }
}

int convertCharToBit(long int n)
{
   FILE* inp = fopen("input.txt", "rb");
   out = fopen("bits.txt", "wb+");
   char ch;
   int i = n * 8;

   while (i)
   {
      ch = fgetc(inp);
      if (ch == -1) {
         break;
```

```
        }
        i--;
        convertToBinary(ch);
    }
    fclose(out);
    fclose(inp);
}

void Encryption(long int plain[])
{
    out = fopen("cipher.txt", "ab+");
    for (int i = 0; i < 64; i++) {
        initialPermutation(i, plain[i]);
    }

    for (int i = 0; i < 32; i++) {
        LEFT[0][i] = IPtext[i];
    }

    for (int i = 32; i < 64; i++) {
        RIGHT[0][i – 32] = IPtext[i];
    }

    for (int k = 1; k < 17; k++)
    {
        cipher(k, 0);

        for (int i = 0; i < 32; i++)
            LEFT[k][i] = RIGHT[k – 1][i];
    }

    for (int i = 0; i < 64; i++)
    {
        if (i < 32) {
            CIPHER[i] = RIGHT[16][i];
        }
        else {
            CIPHER[i] = LEFT[16][i – 32];
        }
        finalPermutation(i, CIPHER[i]);
```

```
    }

    for (int i = 0; i < 64; i++) {
        fprintf(out, "%d", ENCRYPTED[i]);
    }
    fclose(out);
}

void Decryption(long int plain[])
{
    out = fopen("decrypted.txt", "ab+");
    for (int i = 0; i < 64; i++) {
        initialPermutation(i, plain[i]);
    }

    for (int i = 0; i < 32; i++) {
        LEFT[0][i] = IPtext[i];
    }

    for (int i = 32; i < 64; i++) {
        RIGHT[0][i – 32] = IPtext[i];
    }

    for (int k = 1; k < 17; k++)
    {
        cipher(k, 1);

        for (int i = 0; i < 32; i++) {
            LEFT[k][i] = RIGHT[k – 1][i];
        }
    }

    for (int i = 0; i < 64; i++)
    {
        if (i < 32) {
            CIPHER[i] = RIGHT[16][i];
        } else {
            CIPHER[i] = LEFT[16][i – 32];
        }
        finalPermutation(i, CIPHER[i]);
```

```
    }

    for (int i = 0; i < 64; i++) {
        fprintf(out, "%d", ENCRYPTED[i]);
    }

    fclose(out);
}

void convertToBits(int ch[])
{
    int value = 0;
    for (int i = 7; i >= 0; i--) {
        value += (int)pow(2, i) * ch[7 – i];
    }
    fprintf(out, "%c", value);
}

int bittochar()
{
    out = fopen("result.txt", "ab+");
    for (int i = 0; i < 64; i = i + 8) {
        convertToBits(&ENCRYPTED[i]);
    }
    fclose(out);
}

void key56to48(int round, int pos, int text)
{
    int i;
    for (i = 0; i < 56; i++)
    {
        if (PC2[i] == pos + 1) {
            break;
        }
    }
    key48bit[round][i] = text;
}

int key64to56(int pos, int text)
```

```
{
   int i;
   for (i = 0; i < 56; i++)
   {
      if (PC1[i] == pos + 1) {
         break;
      }
   }
   key56bit[i] = text;
}

void key64to48(unsigned int key[])
{
   int k, backup[17][2];
   int CD[17][56];
   int C[17][28], D[17][28];

   for (int i = 0; i < 64; i++) {
      key64to56(i, key[i]);
   }

   for (int i = 0; i < 56; i++)
   {
      if (i < 28) {
         C[0][i] = key56bit[i];
      }
      else {
         D[0][i – 28] = key56bit[i];
      }
   }

   for (int x = 1; x < 17; x++)
   {
      int shift = SHIFTS[x – 1];

      for (int i = 0; i < shift; i++) {
         backup[x - 1][i] = C[x – 1][i];
      }

      for (int i = 0; i < (28 – shift); i++) {
```

```
            C[x][i] = C[x – 1][i + shift];
        }

        k = 0;
        for (int i = 28 – shift; i < 28; i++) {
            C[x][i] = backup[x – 1][k++];
        }

        for (int i = 0; i < shift; i++) {
            backup[x - 1][i] = D[x – 1][i];
        }

        for (int i = 0; i < (28 – shift); i++) {
            D[x][i] = D[x – 1][i + shift];
        }

        k = 0;
        for (int i = 28 – shift; i < 28; i++) {
            D[x][i] = backup[x – 1][k++];
        }
    }

    for (int j = 0; j < 17; j++)
    {
        for (int i = 0; i < 28; i++) {
            CD[j][i] = C[j][i];
        }

        for (int i = 28; i < 56; i++) {
            CD[j][i] = D[j][i – 28];
        }
    }

    for (int j = 1; j < 17; j++)
    {
        for (int i = 0; i < 56; i++) {
            key56to48(j, i, CD[j][i]);
        }
    }
}
```

```
void decrypt(long int n)
{
   FILE* in = fopen("cipher.txt", "rb");
   long int plain[n * 64];
   int i = -1;
   char ch;

   while (!feof(in))
   {
      ch = getc(in);
      plain[++i] = ch – 48;
   }

   for (int i = 0; i < n; i++)
   {
      Decryption(plain + i * 64);
      bittochar();
   }

   fclose(in);
}

void encrypt(long int n)
{
   FILE* in = fopen("bits.txt", "rb");

   long int plain[n * 64];
   int i = -1;
   char ch;

   while (!feof(in))
   {
      ch = getc(in);
      plain[++i] = ch – 48;
   }

   for (int i = 0; i < n; i++) {
      Encryption(plain + 64 * i);
   }
```

```
    fclose(in);
}

void create16Keys()
{
    FILE* pt = fopen("key.txt", "rb");
    unsigned int key[64];
    int i = 0, ch;

    while (!feof(pt))
    {
        ch = getc(pt);
        key[i++] = ch – 48;
    }

    key64to48(key);
    fclose(pt);
}

long int findFileSize()
{
    FILE* inp = fopen("input.txt", "rb");
    long int size;

    if (fseek(inp, 0L, SEEK_END)) {
        perror("fseek() failed");
    }
    // size will contain number of chars in the input file.
    else {
        size = ftell(inp);
    }
    fclose(inp);

    return size;
}

int main()
{
    // destroy contents of these files (from previous runs, if any)
```

```
    out = fopen("result.txt", "wb+");
    fclose(out);

    out = fopen("decrypted.txt", "wb+");
    fclose(out);

    out = fopen("cipher.txt", "wb+");
    fclose(out);

    create16Keys();

    long int n = findFileSize() / 8;
    convertCharToBit(n);

    encrypt(n);
    decrypt(n);

    return 0;
}
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT

EX.NO:8 DATA ENCRYPTION AND DECRYPTION USING RSA (RIVEST, SHAMIR AND ADLEMAN) ALGORITHM

AIM:

To write a c program to perform encryption and decryption using RSA algorithm.

SOFTWARE REQUIRED:

Turbo C, C++

ALGORITHM:

1. Start the program.
2. Transmit the data list by bit.
3. Check for the error.
4. If there is error, correct the error.
5. Stop the program

Program:

```
#include <stdio.h>

int main()
{
  int i, x;
  char str[100];

  printf("\nPlease enter a string:\t");
  gets(str);

  printf("\nPlease choose following options:\n");
  printf("1 = Encrypt the string.\n");
  printf("2 = Decrypt the string.\n");
  scanf("%d", &x);

  //using switch case statements
  switch(x)
  {
```

```
  case 1:
    for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value

    printf("\nEncrypted string: %s\n", str);
    break;

  case 2:
    for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value

    printf("\nDecrypted string: %s\n", str);
    break;

  default:
    printf("\nError\n");
  }
  return 0;
}
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT

EX.NO:9 IMPLEMENTATION OF CLIENT SERVER MODEL USING FTP PROTOCOL

AIM:-

To implement Client Server model using FTP protocol.

SOFTWARE REQUIRED:-

Turbo C, C++

ALGORITHM:

SERVER:

STEP 1: Start
STEP 2: Declare the variables for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Bind the IP address and Port number
STEP 6: Listen and accept the client's request for the connection
STEP 7: Establish the connection with the client
STEP 8: Close the socket
STEP 9: Stop

CLIENT:

STEP 1: Start
STEP 2: Declare the variables for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Call the connect() function
STEP 6: Close the socket
STEP 7: Stop

Program:

SERVER:

```
#include<stdio.h>

#include<arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>
```

```
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
    int sockfd, newsockfd, clength;
    struct sockaddr_in serv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);
    clength=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);
    close(sockfd);
    read(newsockfd, &str, MAX);
```

```
    printf("\nClient message\n File Name : %s\n", str);
    f1=fopen(str, "r");
    while(fgets(buff, 4096, f1)!=NULL)
    {
        write(newsockfd, buff,MAX);
        printf("\n");
    }
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

CLIENT:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(int arg,char*argv[])
{
    int sockfd,n;
    struct sockaddr_in serv_addr;
```

```
    struct hostent*server;
    char send[MAX],recvline[MAX],s[MAX],name[MAX];
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
    printf("\nEnter the source file name : \n");
    scanf("%s",send);
    write(sockfd,send,MAX);
    while((n=read(sockfd,recvline,MAX))!=0)
    {
      printf("%s",recvline);
    }
    close(sockfd);
    return 0;
}
void main()
{
int a[50],b[50];
clrscr();
printf("\nEcho");
printf("\n enter the input address");
scanf("%s",&a);
scanf("%s",&b);
printf("data %s",b);
printf("\n Received data %s",b);
```

```
getch();
}
```

Traceroute :

```
include<stdio.h>
void main()
{
int a,b;
clrscr();
printf("\n ping in client");
printf("\nenter input address");
scanf("%d",&a);
printf("\n communuication established");
printf("\n enter the data");
scanf("%d",&b);
printf("\n server pinging \n server received: %d",b);
getch();
}
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT:-

EX.NO:10 NETWORK TOPOLOGY - STAR, BUS, RING

Aim:

To write a ns2 program for implementing network topology.

Software Required:

NS2 Simulator

Algorithm:

Step 1: start the program.
Step 2: declare the global variables ns for creating a new simulator.
Step 3: set a 'finish' procedure.
Step 4: to create four nodes and create links between the nodes.
Step 5: Create a TCP agent and attach it to node .
Step 6: Create a CBR traffic source and attach it to tcp.
Step 7: create the capable no of nodes.
Step 8: Schedule events for the CBR agents.
Step 9: Call the finish procedure after 5 seconds of simulation time.
Step 10: Run the simulation
Step 11: repeat the procedure for star and ring topology.
Step 12: stop the program.

To monitor traffic for Ring topology using NS2

```
#************************************#

#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
   #Execute nam on the trace file
```

```
    exec nam out.nam &
    exit0
}

#Create four 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/TCP Sink]
$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 packet Size_ 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
```

Aim : To monitor traffic for Bus topology using NS2

```
#*****************************************#

#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
   #Execute nam on the trace file
   exec nam out.nam &
   exit 0
}

#Create four 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/TCP Sink]
$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 packet Size_ 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
```

To monitor traffic for Star topology using NS2

```
#*************************************#

#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 four 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/TCP Sink]
$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 packet Size_ 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
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

Result:

EX.NO:11 TO CREATE SCENARIO AND STUDY THE PERFORMANCE OF NETWORK WITH CSMA / CA PROTOCOL AND COMPARE WITH CSMA/CD PROTOCOLS

Aim:

To write a ns2 program for create scenario and study the performance of network with CSMA / CA protocol and compare with CSMA/CD protocols.

Software Required:

NS2 Simulator

Algorithm:

Step 1: start the program.
Step 2: declare the global variables ns for creating a new simulator.
Step 3: set the color for packets.
Step 4: open the network animator file in the write mode.
Step 5: open the trace file and the win file in the write mode.
Step 6: transfer the packets in network.
Step 7: create the capable no of nodes.
Step 8: create the duplex-link between the nodes including the delay time, bandwidth and dropping queue mechanism.
Step 9: give the position for the links between the nodes.
Step 10: set a tcp connection for source node.
Step 11: set the destination node using tcp sink.
Step 12: set the window size and the packet size for the tcp.
Step 13: set up the ftp over the tcp connection.
Step 14:.set the udp and tcp connection for the source and destination.
Step 15: create the traffic generator CBR for the source and destination files.
Step 15: define the plot window and finish procedure.
Step 16: in the definition of the finish procedure declare the global variables.
Step 17: close the trace file and namefile and execute the network animation file.
Step 18: at the particular time call the finish procedure.
Step 19: stop the program.

Program:

```
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
set fi1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $fi1
set fi2 [open out.nam w]
$ns namtrace-all $fi2
proc finish {} {
global ns fi1 fi2
$ns flush-trace
close $fi1
close $fi2
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetsize_ 552
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
```

```
$ns connect $udp $null
$udp set fid_ 2
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 24.0 "$ftp stop"
$ns at 24.5 "$cbr stop"
proc plotwindow { tcpSource file } {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotwindow $tcpSource $file"
}
$ns at 1.0 "plotwindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
$ns at 125.0 "finish"
$ns run
```

Particulars	**Allotted**	**Obtained**
Performance	50	
Viva voce	10	
Record	15	
Total	75	

Result:

CONTENT BEYOND SYLLABUS

EX.NO:12 SIMULATE A MOBILE ADHOC NETWORK

AIM:

To simulate a Mobile Adhoc network (MANET) using NS2.

SOFTWARE REQUIRED:

Network Simulator -2

ALGORITHM:

1. Create a simulator object
2. Set the values for the parameter
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create the nodes that forms a network numbered from 0 to 3
5. Schedule events and run the program.

Program:

```
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(ifqlen) 50
set val(nn) 3
set val(rp) DSDV
set ns [new Simulator]
set tf [open output.tr w]
$ns trace-all $tf
set tf1 [open output.nam w]
$ns namtrace-all-wireless $tf1 100 100
set topo [new Topography]
$topo load_flatgrid 100 100
create-god $val(nn)
$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) \
-channelType $val(chan) \
```

```
-topoInstance $topo \
-agentTrace ON \
-routerTrace OFF \
-macTrace OFF \
-movementTrace OFF
set node0 [$ns node]
set node1 [$ns node]
set node2 [$ns node]
$ns initial_node_pos $node0 10
$ns initial_node_pos $node1 10
$ns initial_node_pos $node2 10
$node0 set X_ 25.0
$node0 set Y_ 50.0
$node0 set Z_ 0.0
$node1 set X_ 50.0
$node1 set Y_ 50.0
$node1 set Z_ 0.0
$node2 set X_ 65.0
$node2 set Y_ 50.0
$node2 set Z_ 0.0
set tcp1 [new Agent/TCP]
$ns attach-agent $node0 $tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $node2 $sink1
$ns connect $tcp1 $sink1
$ns at 10.0 "$node1 setdest 50.0 90.0 0.0"
$ns at 50.0 "$node1 setdest 50.0 10.0 0.0"
$ns at 0.5 "$ftp start"
$ns at 1000 "$ftp stop"
$ns at 1000 "finish"
proc finish {} {
global ns tf tf1
$ns flush-trace
close $tf
exec nam output.nam &
exit 0
}
$ns run
```

Particulars	**Allotted**	**Obtained**
Performance	**50**	
Viva voce	**10**	
Record	**15**	
Total	**75**	

RESULT

EX.NO:13 IMPLEMENT TRANSPORT CONTROL PROTOCOL IN SENSOR NETWORK

AIM:

To implement a Transport Control Protocol in sensor network through the simulator.

SOFTWARE REQUIRED:

Network Simulator -2

ALGORITHM:

1.Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between n(0) and n(4)
7. Apply FTP Traffic over TCP
8. Setup UDP Connection between n(1) and n(5)
9. Apply CBR Traffic over UDP.
10. Schedule events and run the program.

Program:

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#Define a 'finish' procedure
proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
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]
$n1 color red
$n1 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
# PPP
$ns at 125.0 "finish"
$ns run
```

Particulars	Allotted	Obtained
Performance	50	
Viva voce	10	
Record	15	
Total	75	

RESULT