KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2018 onwards)

DEPARTMENT OF COMPUTER SCIENCE,CA& IT

SUBJECT :COMPUTER NETWORKS– PRACTICAL	**SEMESTER**: III	**L T P**
SUBJECT CODE:18CSU313	**CLASS:IIB.Sc.CS–A**	**0 0 4**

Course Objectives

- To master the fundamentals of data communications networks by gaining a working knowledge of data transmission concepts,
- To understand the operation of all seven layers of OSI Model and the protocols used in each layer.

Course Outcomes(COs)

1. Various transmission media, their comparative study, fiber optics and wireless media
2. Categories and topologies of networks (LAN and WAN) Layered architecture (OSI and TCP/IP) and protocol suites.
3. Channel error detection and correction, MAC protocols, Ethernet and WLAN.
4. Details of IP operations in the INTERNET and associated routing principles

LIST OF PACTICAL

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.

STAFF **HOD**

Ex No : 1 Date :	CYCLIC REDUNDANCY CHECK (CRC) ERROR DETECTION ALGORITHM

AIM:

To simulate the cyclic redundancy check (CRC) error detection algorithm.

ALGORITHM:

Step 1: Start the program

Step 2: Accept the number of bits in the input data stream (t).

Step 3: Set the number of bits in the divisor (g).

Step 4: Accept the input bits and the divisor bits.

Step 5: Append (g-3) '0' bits to the end of the input data bits.

Step 6: Perform binary division

Step 7: Remove the appended '0' bits and append the remainder bits to the original data bits.

Step 8: Again perform binary division.

Step 9: If the reminder in zero then there is no error in the transmitted bits else the received bits are in error.

Step 10: Stop the program

PROGRAM :

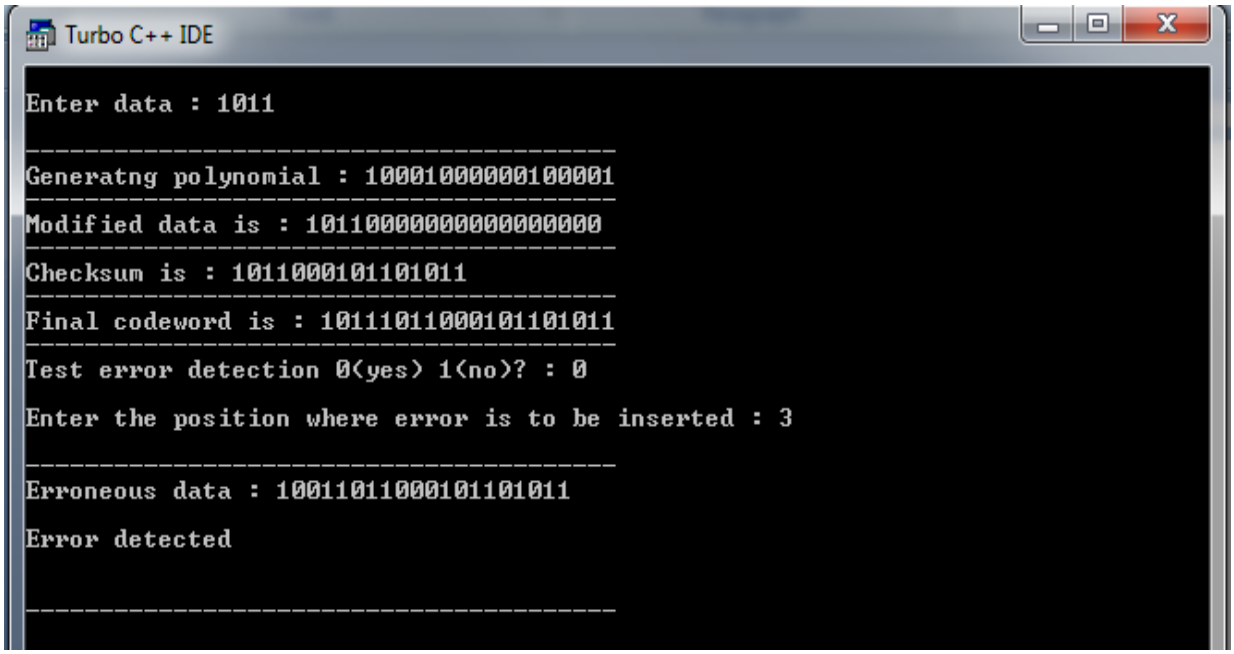
```
#include<stdio.h>
#include<string.h>
#define N strlen(g)

char t[28],cs[28],g[]="10001000000100001";
int a,e,c;
void xor()
{
   for(c = 1;c < N; c++)
   cs[c] = (( cs[c] == g[c])?'0':'1');
}
void crc()
{
   for(e=0;e<N;e++)
      cs[e]=t[e];
   do{
      if(cs[0]=='1')
         xor();
      for(c=0;c<N-1;c++)
         cs[c]=cs[c+1];
      cs[c]=t[e++];
   }
while(e<=a+N-1);
}
```

```
int main()
{
   printf("\nEnter data : ");
   scanf("%s",t);
   printf("\n---------------------------------------");
   printf("\nGeneratng polynomial : %s",g);
   a=strlen(t);
   for(e=a;e<a+N-1;e++)
      t[e]='0';
   printf("\n---------------------------------------");
   printf("\nModified data is : %s",t);
   printf("\n---------------------------------------");
   crc();
   printf("\nChecksum is : %s",cs);
   for(e=a;e<a+N-1;e++)
      t[e]=cs[e-a];
   printf("\n---------------------------------------");
   printf("\nFinal codeword is : %s",t);
   printf("\n---------------------------------------");
   printf("\nTest error detection 0(yes) 1(no)? : ");
   scanf("%d",&e);
   if(e==0)
   {
      do{
         printf("\nEnter the position where error is to be inserted : ");
         scanf("%d",&e);
```

```
        }while(e==0 || e>a+N-1);
        t[e-1]=(t[e-1]=='0')?'1':'0';
        printf("\n----------------------------------------");
        printf("\nErroneous data : %s\n",t);
    }
    crc();
    for(e=0;(e<N-1) && (cs[e]!='1');e++);
        if(e<N-1)
            printf("\nError detected\n\n");
        else
            printf("\nNo error detected\n\n");
            printf("\n----------------------------------------\n");
        return 0;
}
```

Output:


Turbo C++ IDE
Enter data : 1011
Generatng polynomial : 10001000000100001
Modified data is : 10110000000000000000
Checksum is : 1011000101101011
Final codeword is : 10111011000101101011
Test error detection 0(yes) 1(no)? : 0
Enter the position where error is to be inserted : 3
Erroneous data : 10011011000101101011
Error detected

Ex No : 2 Date :	STOP AND WAIT PROTOCOL

AIM:

To simulate and implement stop and wait protocol for noisy channel.

ALGORITHM:

Step 1: Start the process

Step 2: Declare all necessary variables.

Step 3: The number of frames to be sent is generated using rand() function.

Step 4: The sender sends one data packet at a time and sends next packet only after receiving acknowledgement for previous.

Step 5: Using the sleep function acknowledgement is paused for few seconds.

Step 6: The stopped frames are retransmitted and acknowledgment is received for those frames.

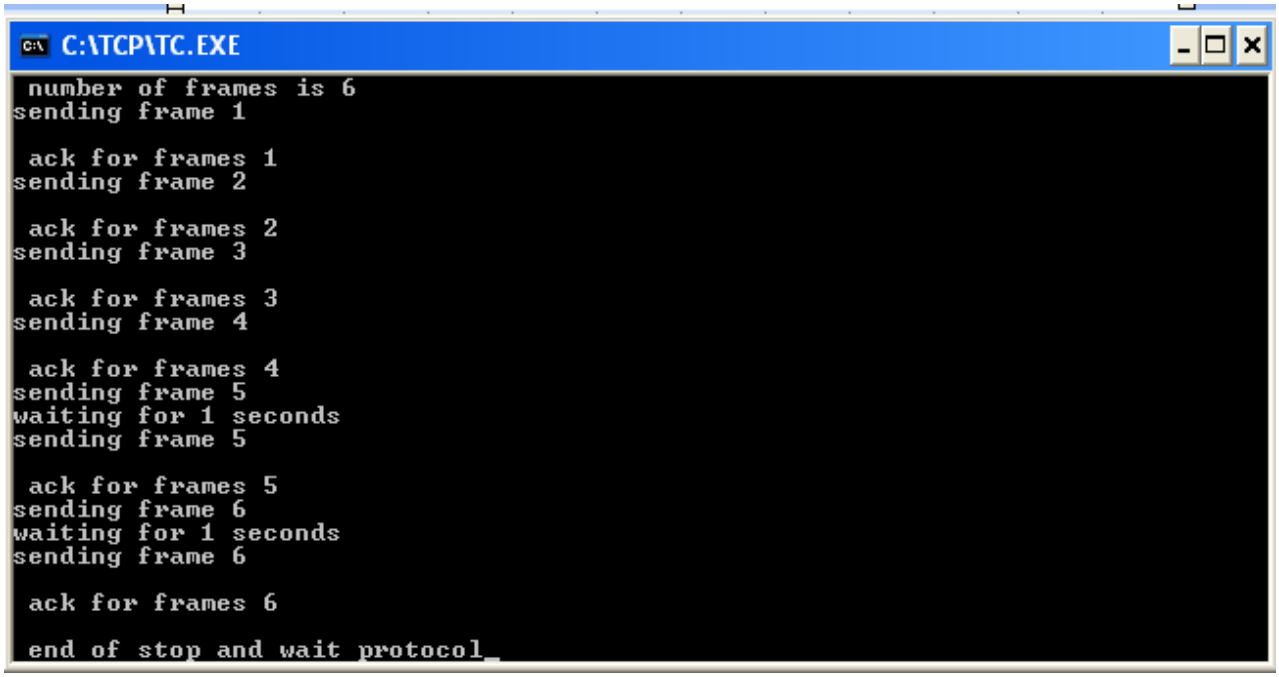
Step 7: After all frames are sent stop the process.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int i,j,noframes,x,x1=10,x2;
clrscr();
for(i=0;i<200;i++)
rand();
noframes=rand()/200;
i=1;
j=1;
noframes=noframes/8;
printf("\n number of frames is %d",noframes);
getch();
        while(noframes>0)
        {
        printf("\n sending frame %d",i);
        srand(x1++);
        x=rand()%10;
        if(x%2==0)
        {
        for(x2=1;x2<2;x2++)
        {
        printf("waiting for %d seconds \n",x2);
```

```
        sleep(x2);
        }
        printf("\n sending frame %d",i);
        srand(x1++);
        x=rand()%10;
        }
printf("\n ack for frames %d",j);
noframes-=1;
i++;
j++;
}
printf("\n end of stop and wait protocol");
getch();
}
```

OUTPUT :


C:\TCP\TC.EXE
number of frames is 6
sending frame 1
ack for frames 1
sending frame 2
ack for frames 2
sending frame 3
ack for frames 3
sending frame 4
ack for frames 4
sending frame 5
waiting for 1 seconds
sending frame 5
ack for frames 5
sending frame 6
waiting for 1 seconds
sending frame 6
ack for frames 6
end of stop and wait protocol

Ex No : 3 Date :	GO BACK N SLIDING WINDOW PROTOCOL

AIM:

To simulate and implement go back n sliding window protocol.

ALGORITHM:

Step 1: Start the process

Step 2: Declare all necessary variables.

Step 3: Define the window Size by getting input from the user.

Step 4: Sender transmits all frames present in the window.

Step 5: When acknowledgment is not received for a frame, then the sender retransmits the frames from the last transmitted frame.

Step 6: After all frames are sent successfully stop the process.

PROGRAM :

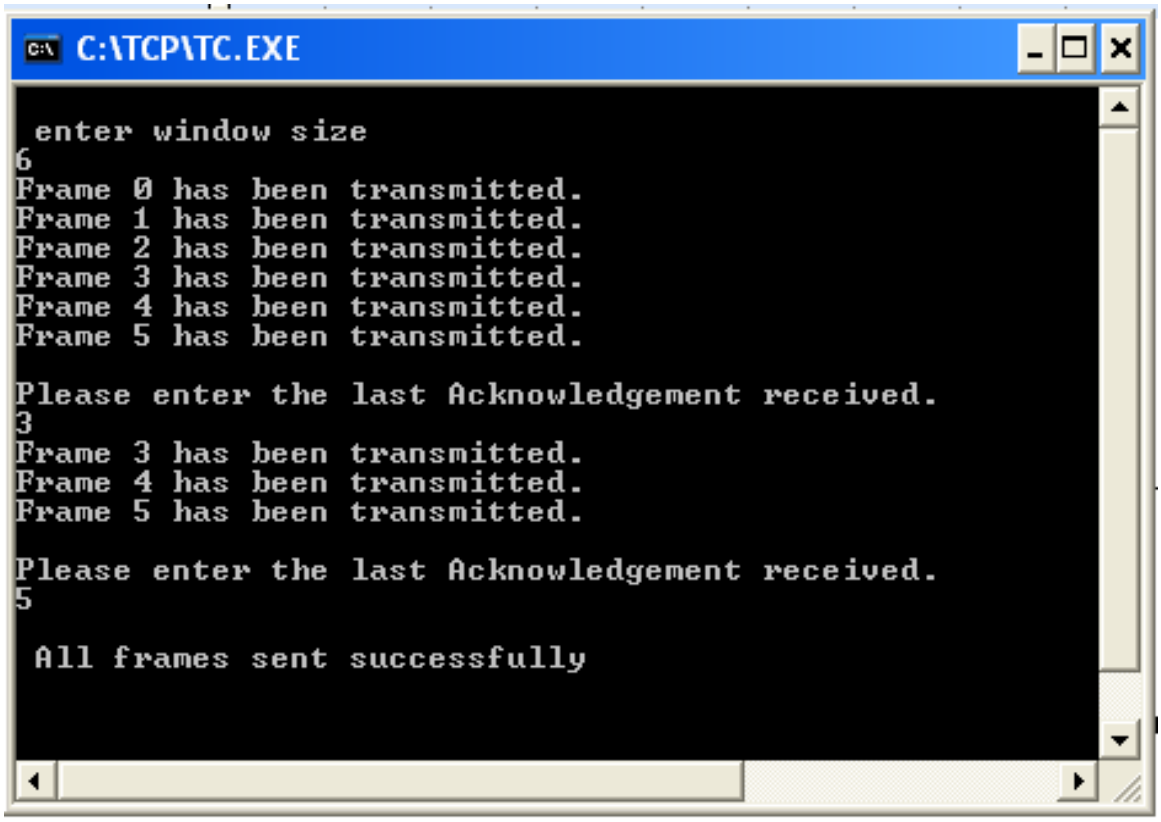
```
#include<stdio.h>
#include<conio.h>
int main()
{
        int windowsize,sent=0,ack,i;
        clrscr();
        printf("enter window size \n");
        scanf("%d",&windowsize);
        while(1)
        {
                for( i = 0; i < windowsize; i++)
                        {
                                printf("Frame %d has been transmitted.\n",sent);
                                sent++;
                                if(sent == windowsize)
                                   break;

                        }
                        printf("\nPlease enter the last Acknowledgement
received.\n");
                        scanf("%d",&ack);

                        if(ack == windowsize-1)
                        {
```

```
                printf("\n All frames sent successfully");
                getch();
                break;
                }
                else
                        sent = ack;
        }
return 0;
}
```

OUTPUT :


C:\TCP\TC.EXE
enter window size
6
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Please enter the last Acknowledgement received.
3
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Please enter the last Acknowledgement received.
5
All frames sent successfully

Ex No : 4 **Date :**	**SELECTIVE REPEAT SLIDING WINDOW PROTOCOL**

AIM:

To simulate and implement selective repeat sliding window protocol

ALGORITHM:

Step 1: Start the process

Step 2: Declare all necessary variables.

Step 3: Define the functions recvfrm(void), resend(void), selective(void).

Step 4: In the sender function enter the number of packets and data to be sent.

Step 5: In recvfrm function get the packet number that is not received using rand() function.

Step 6: In the resend() function the packet for which acknowledgment is not received is resent.

Step 7: After all frames are sent successfully stop the process

PROGRAM :

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int n,r;
struct frame
{
char ack;
int data;
}frm[10];

int sender(void);
void recvfrm(void);
void resend(void);
void selective(void);

void  main()
{
    clrscr();
    printf("\nSelective repeat\n\n ");
    selective();
    getch();
}

void selective()
```

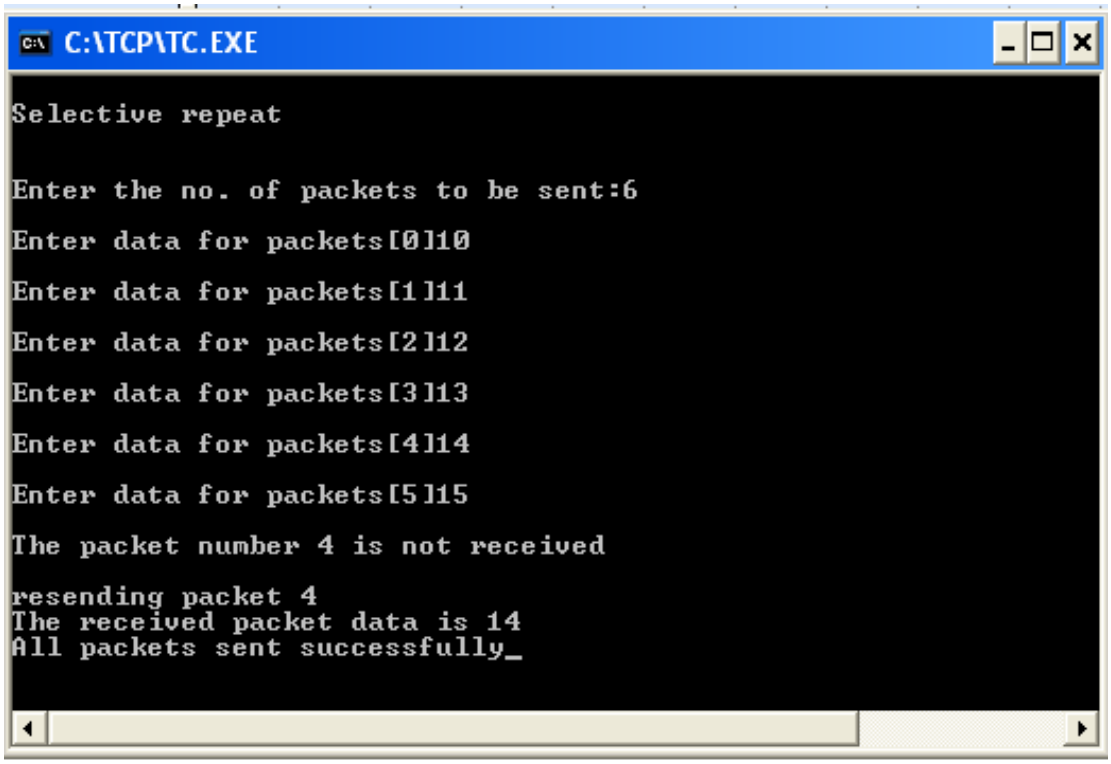
```
{
sender();
recvfrm();
resend();
printf("\nAll packets sent successfully");
}

int sender()
{
int i;
printf("\nEnter the no. of packets to be sent:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter data for packets[%d]",i);
scanf("%d",&frm[i].data);
frm[i].ack='y';
}
return 0;
}

void recvfrm()
{
int i;
rand();
r=rand()%n;
frm[r].ack='n';
```

```
for(i=0;i<n;i++)
{
 if(frm[i].ack=='n')
 printf("\nThe packet number %d is not received\n",r);
}
}
void resend()
{
      printf("\nresending packet %d",r);
      sleep(2);
      frm[r].ack='y';
      printf("\nThe received packet data is %d",frm[r].data);
}
```

OUTPUT :


C:\TCP\TC.EXE
Selective repeat
Enter the no. of packets to be sent:6
Enter data for packets[0]10
Enter data for packets[1]11
Enter data for packets[2]12
Enter data for packets[3]13
Enter data for packets[4]14
Enter data for packets[5]15
The packet number 4 is not received
resending packet 4
The received packet data is 14
All packets sent successfully_

Ex No : 5 **Date :**	**DISTANCE VECTOR ROUTING ALGORITHM**

AIM:

To simulate and implement distance vector routing algorithm

ALGORITHM:

Step 1: Start the program

Step 2: Assign the distance of the node to zero.

Step 3: Accept the input distance matrix from which represents the distance between each node in the network.

Step 4: Store the distance between nodes in a suitable variable.

Step 5: Calculate the minimum distance between two nodes by iterating.

Step 6: If the distance between two nodes is larger than the calculated alternate available path, replace the existing distance with the calculated distance.

Step 7: Print the shortest path calculated.

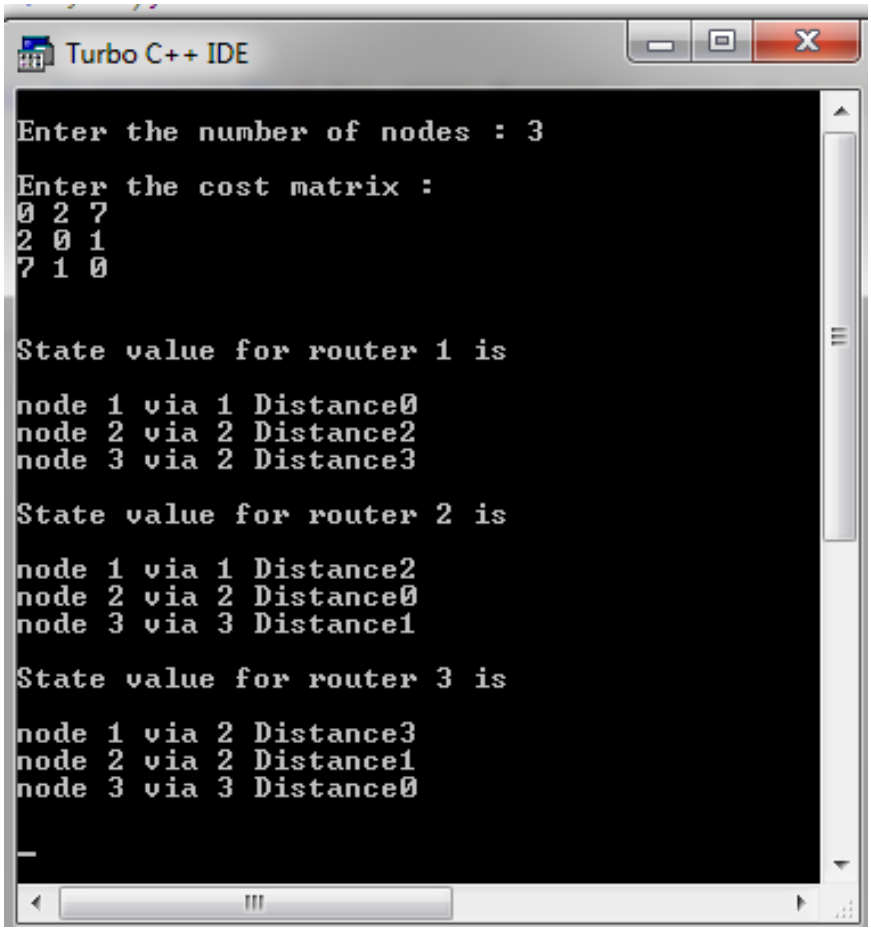
Step 8: Stop the program.

PROGRAM

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
void 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");
getch();
}
```

OUTPUT :


Turbo C++ IDE
Enter the number of nodes : 3
Enter the cost matrix :
0 2 7
2 0 1
7 1 0
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
node 3 via 2 Distance3
State value for router 2 is
node 1 via 1 Distance2
node 2 via 2 Distance0
node 3 via 3 Distance1
State value for router 3 is
node 1 via 2 Distance3
node 2 via 2 Distance1
node 3 via 3 Distance0

Ex No : 6 **Date :**	**DIJKSTRA ALGORITHM FOR SHORTEST PATH ROUTING**

AIM:

To simulate and implement Dijkstra algorithm for shortest path routing.

ALGORITHM:

Step 1: Start the process.

Step 2: Include all the header files.

Step 3: Declare all necessary variables and function for shortest path().

Step 4: Get the value for no. of vertices.

Step 5: Get the value for no. of edges.

Step 6: Get the initial vertex for which distance is to be calculated.

Step 7: Calculate the shortest path using minimum distance.

Step 8: Display the output.

Step 9: Stop the process

PROGRAM :

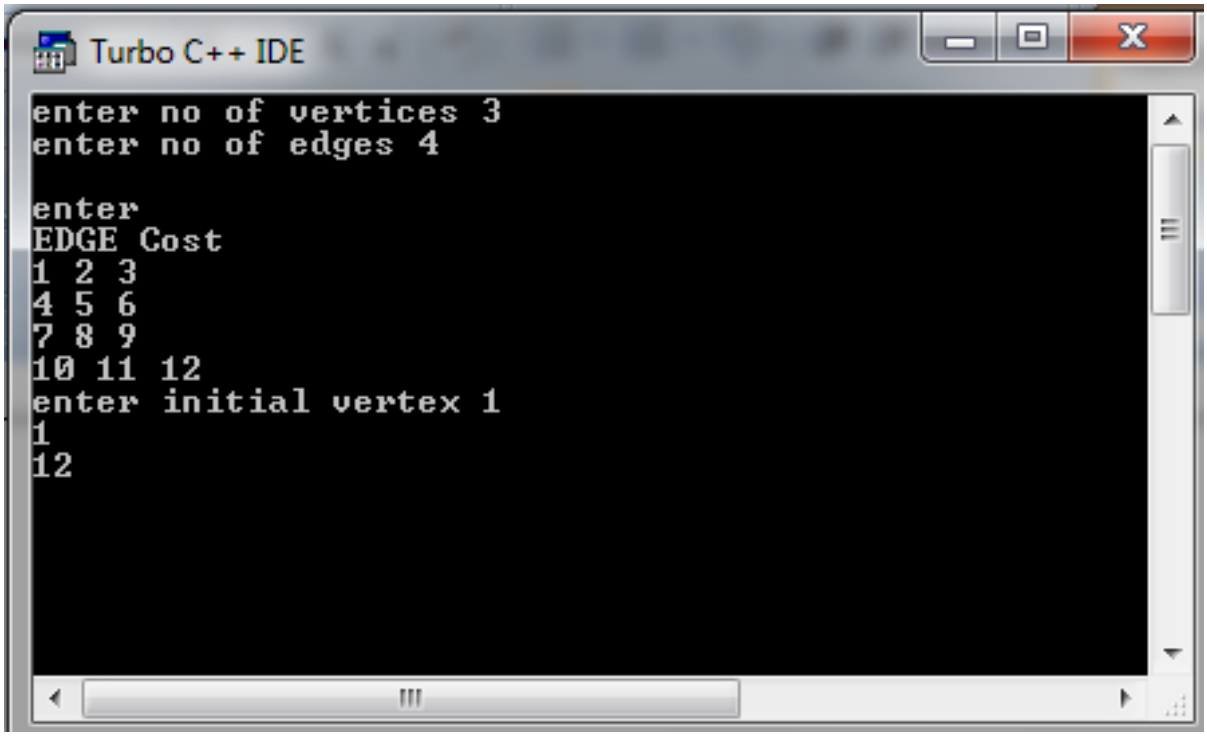
```
#include<iostream.h>
#include<conio.h>
int shortest(int ,int);
int
cost[10][10],dist[20],i,j,n,k,m,S[20],v,totcost,pat
h[20],p;
main()
{
int c;
cout <<"enter no of vertices";
cin >> n;
cout <<"enter no of edges";
cin >>m;
cout <<"\nenter\nEDGE Cost\n";
for(k=1;k<=m;k++)
{
```

```
cin >> i >> j >>c;
cost[i][j]=c;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]==0)
cost[i][j]=31999;
cout <<"enter initial vertex";
cin >>v;
cout << v<<"\n";
shortest(v,n);
}
int shortest(int v,int n)
{
int min;
for(i=1;i<=n;i++)
{
```

```
S[i]=0;
dist[i]=cost[v][i];
}
path[++p]=v;
S[v]=1;
dist[v]=0;
for(i=2;i<=n-1;i++)
{
k=-1;
min=31999;
for(j=1;j<=n;j++)
{
if(dist[j]<min && S[j]!=1)
{
min=dist[j];
k=j;
}
```

```
}
if(cost[v][k]<=dist[k])
p=1;
path[++p]=k;
for(j=1;j<=p;j++)
cout<<path[j];
cout <<"\n";
//cout <<k;
S[k]=1;
for(j=1;j<=n;j++)
if(cost[k][j]!=31999 &&
dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
dist[j]=dist[k]+cost[k][j];
}
}
```

OUTPUT :


Turbo C++ IDE
enter no of vertices 3
enter no of edges 4
enter
EDGE Cost
1 2 3
4 5 6
7 8 9
10 11 12
enter initial vertex 1
1
12