INDEX

Experiment No: 1 (a)

NAME OF THE EXPERIMENT: Bit Stuffing.

AIM: Write a C program to implement the data link layer framing methods such as bit stuffing.

Source Code:

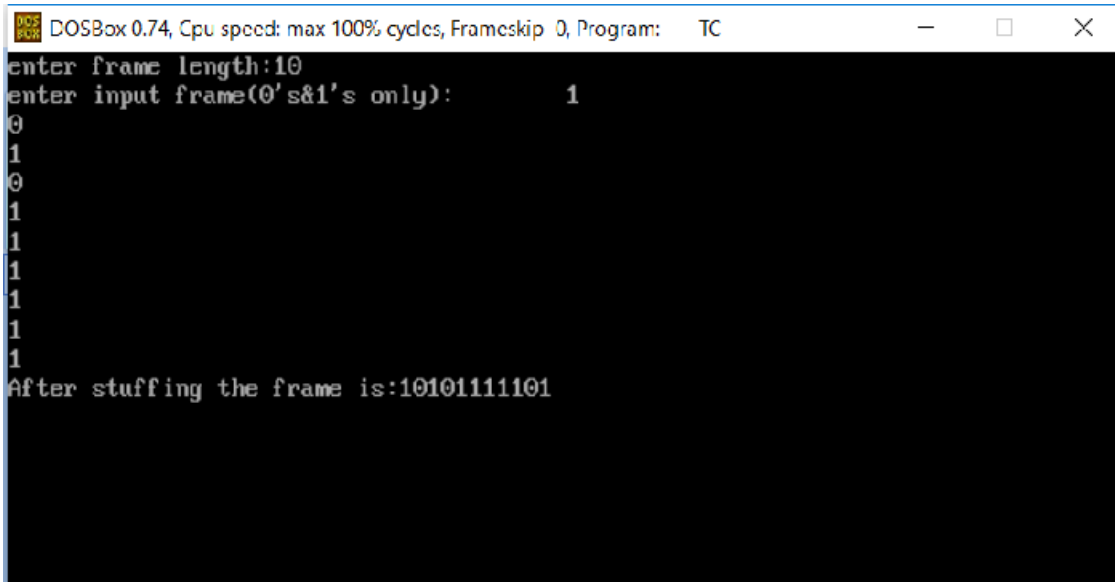
```
#include<stdio.H>
#include<conio.H>
#include<string.h>
void main()
{
  int a[20],b[30],i,j,k,count,n;
  clrscr();
  printf("enter frame length:");
  scanf("%d",&n);
  pritf("enter input frame(0's&1's
only):");
  for(i=0;i<n;i++)
  scanf("%d",&a[i]);
  i=0;count=1;j=0;
  while(i<n)
  {
   if(a[i]==1)
  {
   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 stuffing the frame is:");
 for(i=0;i<j;i++)
printf("%d",b[i]);
getch();
}
```

Output:

Enter the number of bits:
10
1
0
1
0
1
1
1
1
1
1
Data after stuffing: 10101111101

OUTPUT CONSOLE:



Experiment No: 1(b)

NAME OF THE EXPERIMENT: Character Stuffing.

AIM: Write a C program to Implement the data link layer framing method such as character stuffing.

Source Code:

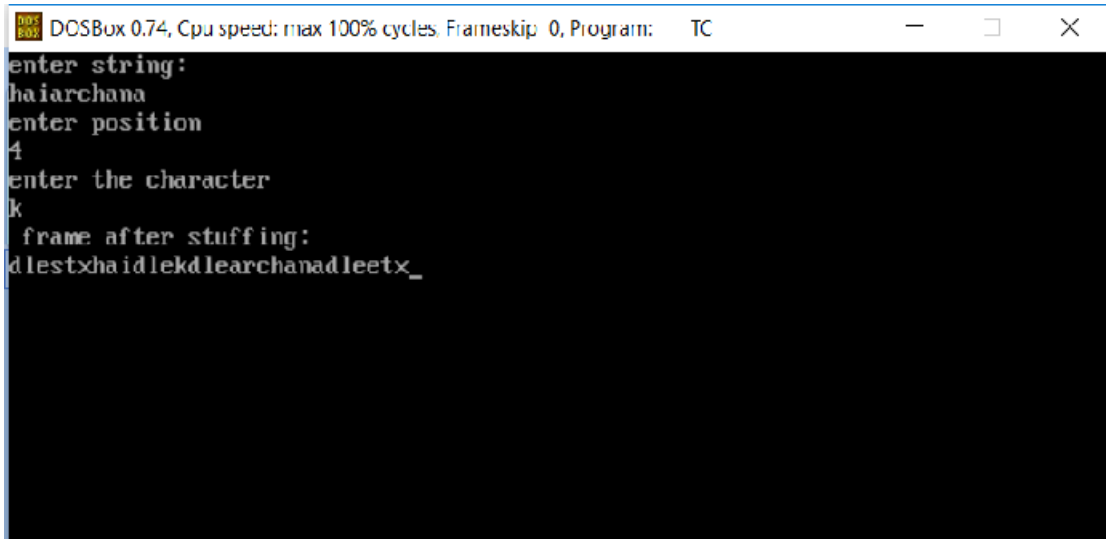
```
//program for character stuffing
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<process.h>

void main()
{
 int i=0,j=0,n,pos;
 char a[20],b[50],ch;
 clrscr();
 printf("enter string:\n");
 scanf("%s",&a);
 n=strlen(a);
 printf("enter position\n");
 scanf("%d",&pos);
 if(pos>n)
{
  printf("invalid position,Enter again:");
  scanf("%d",&pos);
}
printf("enter the character\n");
ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
 if(i==pos-1)
 {
  b[j]='d';
  b[j+1]='l';
  b[j+2]='e';
  b[j+3]=ch;
```

```
  b[j+4]='d';

  b[j+5]='l';
  b[j+6]='e';
  j=j+7;
 }
if(a[i]=='d'&&a[i+1]=='l'&& a[i+2]=='e')
{
 b[j]='d';
 b[j+1]='l';
 b[j+2]='e;
 j=j+3;
}
b[j]=a[i];
i++;
j++;
}
  b[j]='d';
  b[j+1]='l';
  b[j+2]='e';
  b[j+3]='e';
  b[j+4]='t';
  b[j+5]='x';
  b[j+6]='\0';
printf("\n frame after stuffing: \n");
printf("%s",b);
getch();
 }
```

OUTPUT:

```
Enter String:
haiarchana
Enter position:
4
Enter the Character
K
Frame after stuffing:
dlestxhaidlekdlearchanadleetx
```

OUTPUT CONSOLE:



Experiment No: 1(c)

NAME OF THE EXPERIMENT: Character Count

AIM: Write a C program to implement data link layer framing method character count.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char data[20][20];
int n;
void main()
{
int i,ch,j;
char tmp[20][20];
clrscr();
printf("Enter the number of frames:");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
if(i!=0)
{
printf("frame%d:",i);
fflush(stdin);
gets(data[i]);
}
}
/*saving frame with count and data*/
for(i=0;i<=n;i++)
{
tmp[i][0]=49+strlen(data[i]);
tmp[i][1]='\0';
```

```
strcat(tmp[i],data[i]);
}
printf("\n\t\tAT THE SENDER:\n");

printf("Data as frames:\n");
for(i=1;i<=n;i++)
{
printf("Frame%d:",i);
puts(tmp[i]);
}
printf("Data transmitted:");
for(i=1;i<=n;i++)
printf("%s",tmp[i]);
printf("\n\t\tAT THE RECEIVER\n");
printf("The data received:");
for(i=1;i<=n;i++)
{
ch=(int)(tmp[i][0]-49);
for(j=1;j<=ch;j++)
data[i][j-1]=tmp[i][j];
data[i][j-1]='\0';
}
printf("\n The data after removing count char:");
for(i=1;i<=n;i++)
printf("%s",data[i]);
printf("\n The data in frame form:\n");
for(i=1;i<=n;i++)
{
printf("Frame%d:",i);
puts(data[i]);
}
```

```
getch();
}
```

OUTPUT:

Enter the no. Of frames: 2

Frame1: computer

Frame2: networks

AT THE SENDER

Data as frames:

Frame1:9computer

Frame2:9networks

Data transmitted :9computer9networks

AT THE RECEIVER

The data received.

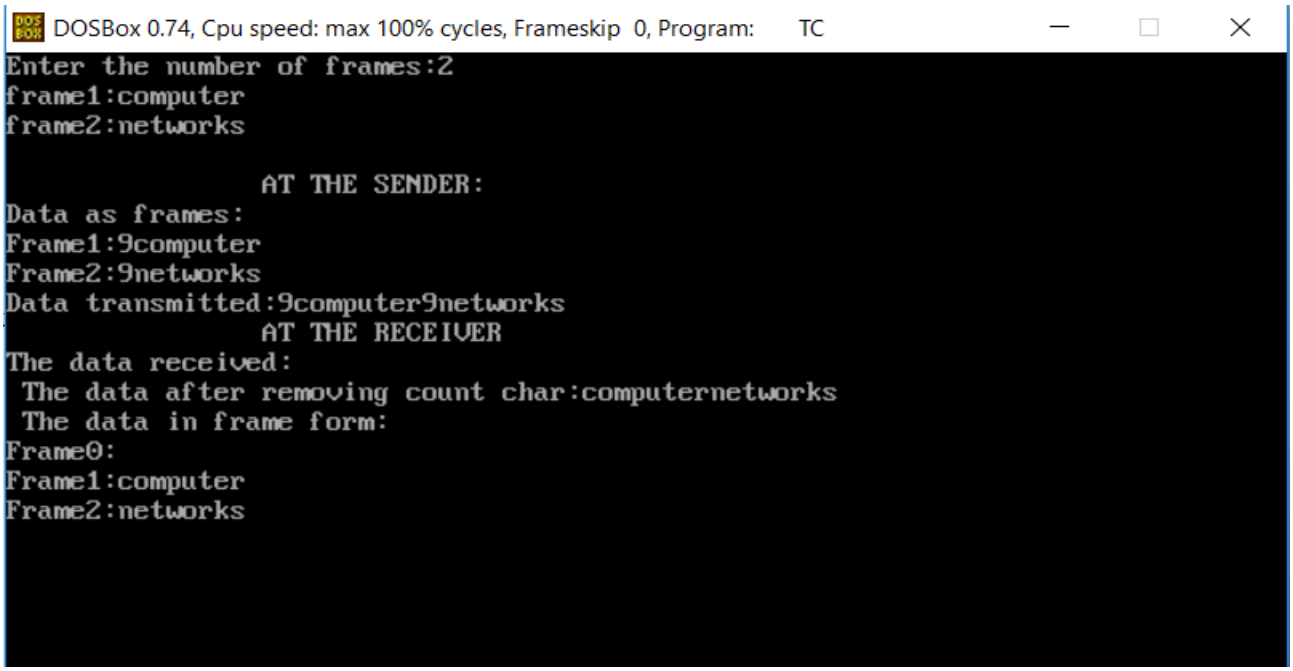
The data after removing count char: computer networks

The data in frame form:

Frame1: computer

Frame2: networks

OURPUT CONSOLE:



Experiment No: 2

NAME OF THE EXPERIMENT: Cyclic Redundancy Check

AIM: Write a C program to implement on a data set characters the three CRC polynomials – CRC 12, CRC 16, and CRC CCIP.

Source Code:

// program for Cyclic Redundancy Check

```
#include<stdio.h>
#include<conio.h>
int main(void)
{
int data[50],div[16],rem[16];
int datalen, divlen, i,j,k;
int ch;
clrscr();
printf("Enter the data: ");
i = 0;
while((ch = fgetc(stdin)) != '\n')
{
```

```
if(ch == '1')
data[i] = 1;
else
data[i] = 0;
i++;
}
datalen = i;
printf("\nEnter the divisor: ");
i = 0;
while((ch = fgetc(stdin)) != '\n')
{
if(ch == '1')
div[i] = 1;
else
div[i] = 0;

i++;
}
divlen = i;
for(i = datalen ; i < datalen + divlen - 1 ; i++)
data[i] = 0;
datalen = datalen + divlen - 1;
for(i = 0 ; i < divlen ; i++)
rem[i] = data[i];
k = divlen-1;
while(k < datalen)
if(rem[0] == 1)
{
for(i = 0 ; i < divlen ; i++)
rem[i] = rem[i] ^ div[i];
}
```

```
else
{
if(k == datalen-1)
break;
for(i = 0 ; i < divlen-1 ; i++)
{
rem[i] = rem[i+1];
printf("%d",rem[i]);
}
rem[i] = data[++k];
printf("%d\n",rem[i]);
}
j=1;
for(i = datalen - divlen + 1 ; i < datalen ; i++)
{
data[i] = rem[j++];
}
printf("\nThe data to be sent is\n");
for(i = 0 ; i < datalen ; i++)
printf("%d",data[i]);
getch();
return 0;
}
```

OUTPUT:

```
Enter the data: 10101111
Enter the divisor: 1011
0011
0111
1111
1001
0100
```

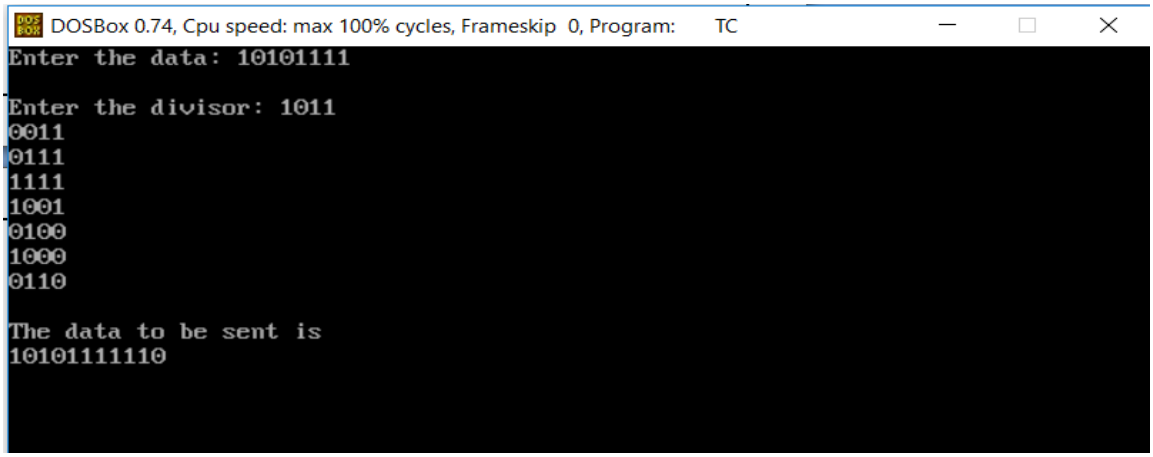
1000

0110

The data to be sent is

10101111110

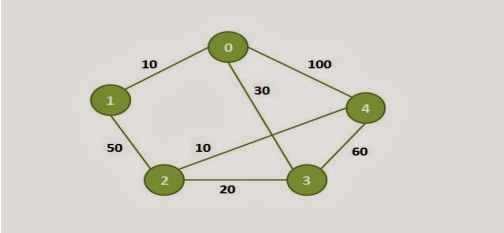
OUTPUT CONSOLE:



Experiment No: 3

NAME OF THE EXPERIMENT: Shortest path

AIM: Write a C program to Implement Dijkstra's Algorithm to compute the shortest path through a given path.



SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
   int G[MAX][MAX],i,j,n,u;
   printf("Enter no. of vertices:");
```

```
    scanf("%d",&n);
    printf("\nEnter the adjacency matrix:\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);
    printf("\nEnter the starting node:");
    scanf("%d",&u);
    dijkstra(G,n,u);
    return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int visited[MAX],count,mindistance,nextnode,i,j;
    //pred[] stores the predecessor of each node
    //count gives the number of nodes seen so far
    //create the cost matrix
 for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(G[i][j]==0)
                cost[i][j]=INFINITY;
            else
                cost[i][j]=G[i][j];
    //initialize pred[],distance[] and visited[]
    for(i=0;i<n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
    distance[startnode]=0;
    visited[startnode]=1;
    count=1;
    while(count<n-1)
    {
        mindistance=INFINITY;
```

```
            //nextnode gives the node at minimum distance
        for(i=0;i<n;i++)
            if(distance[i]<mindistance&&!visited[i])
            {
                mindistance=distance[i];
                nextnode=i;
            }
            //check if a better path exists through nextnode
            visited[nextnode]=1;
            for(i=0;i<n;i++)
                if(!visited[i])
                    if(mindistance+cost[nextnode][i]<distance[i])
                    {
                        distance[i]=mindistance+cost[nextnode][i];
             pred[i]=nextnode;
                    }
        count++;
    }
     //print the path and distance of each node
    for(i=0;i<n;i++)
        if(i!=startnode)
        {
            printf("\nDistance of node%d=%d",i,distance[i]);
            printf("\nPath=%d",i);
            j=i;
            do
            {
                j=pred[j];
                printf("<-%d",j);
            }while(j!=startnode);
    }
}
```

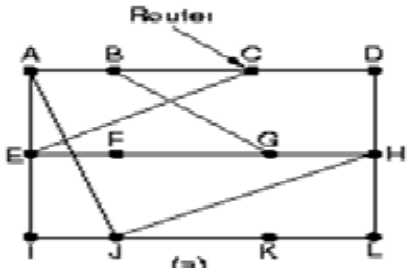
OUTPUT CONSOLE:

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Program:    TC
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0_
```

Experiment No: 4

NAME OF THE EXPERIMENT: Distant Vector Routing

AIM: Write a C program to take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table art each node using distance vector routing algorithm.



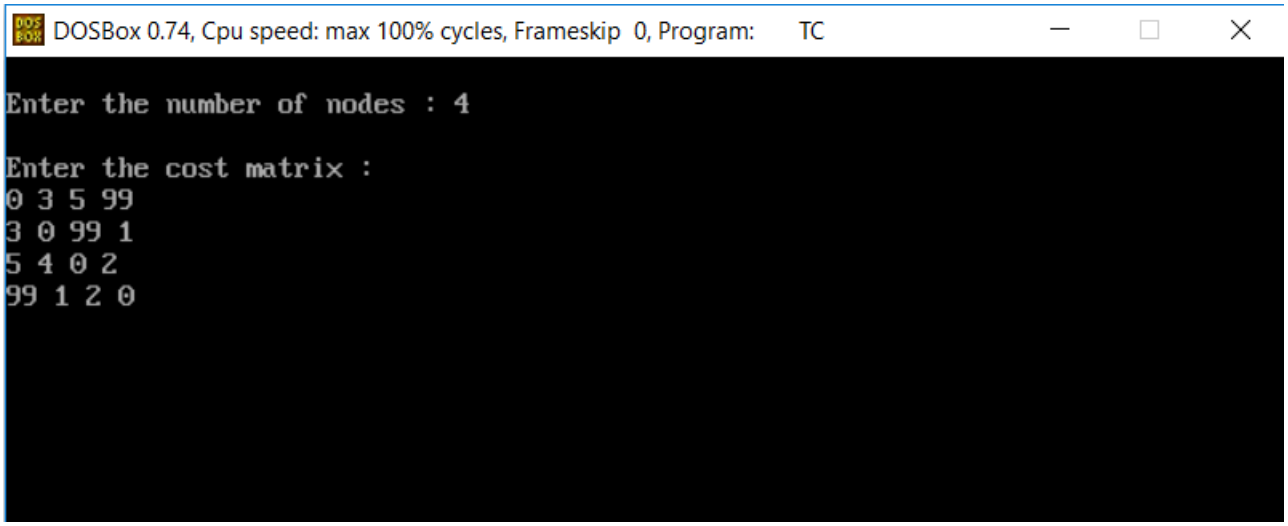
Source Code:

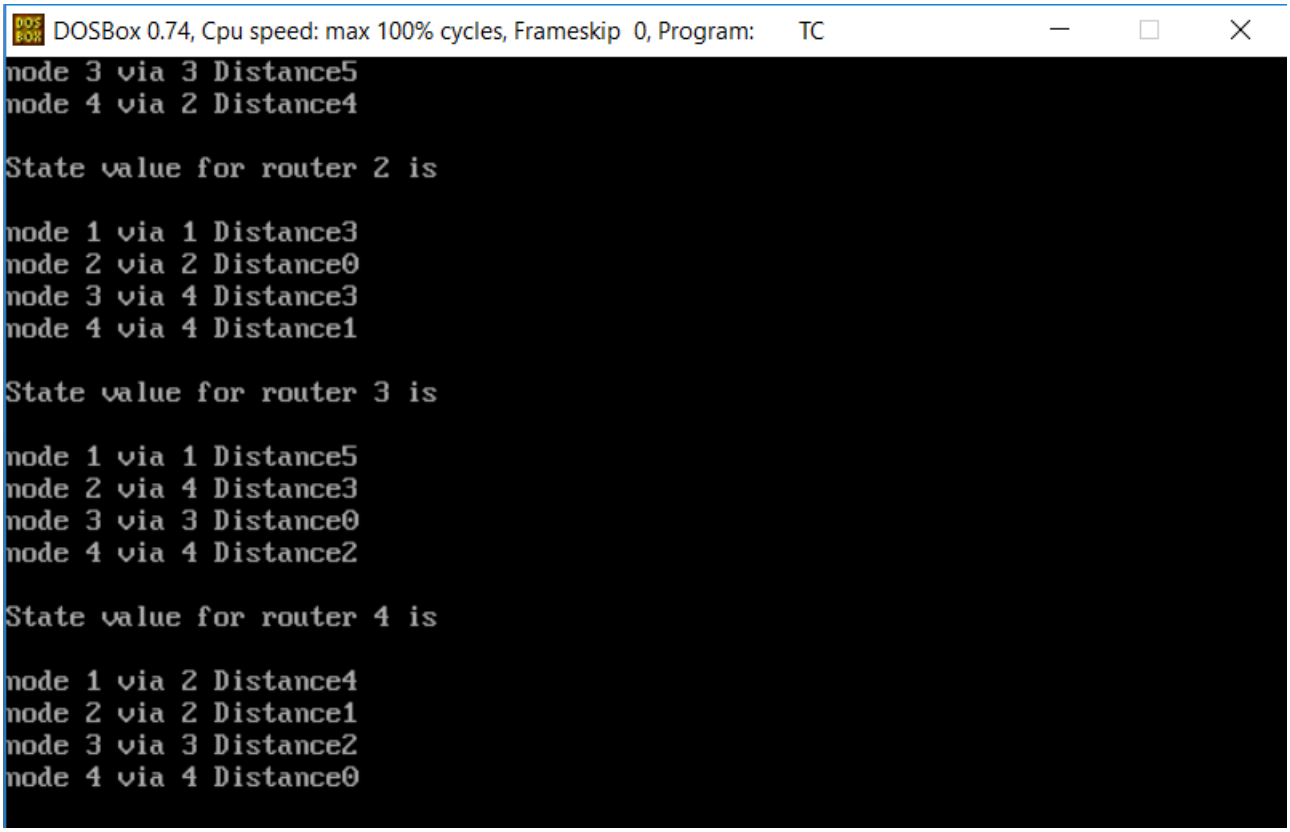
```
#include<stdio.h>

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");
}
```

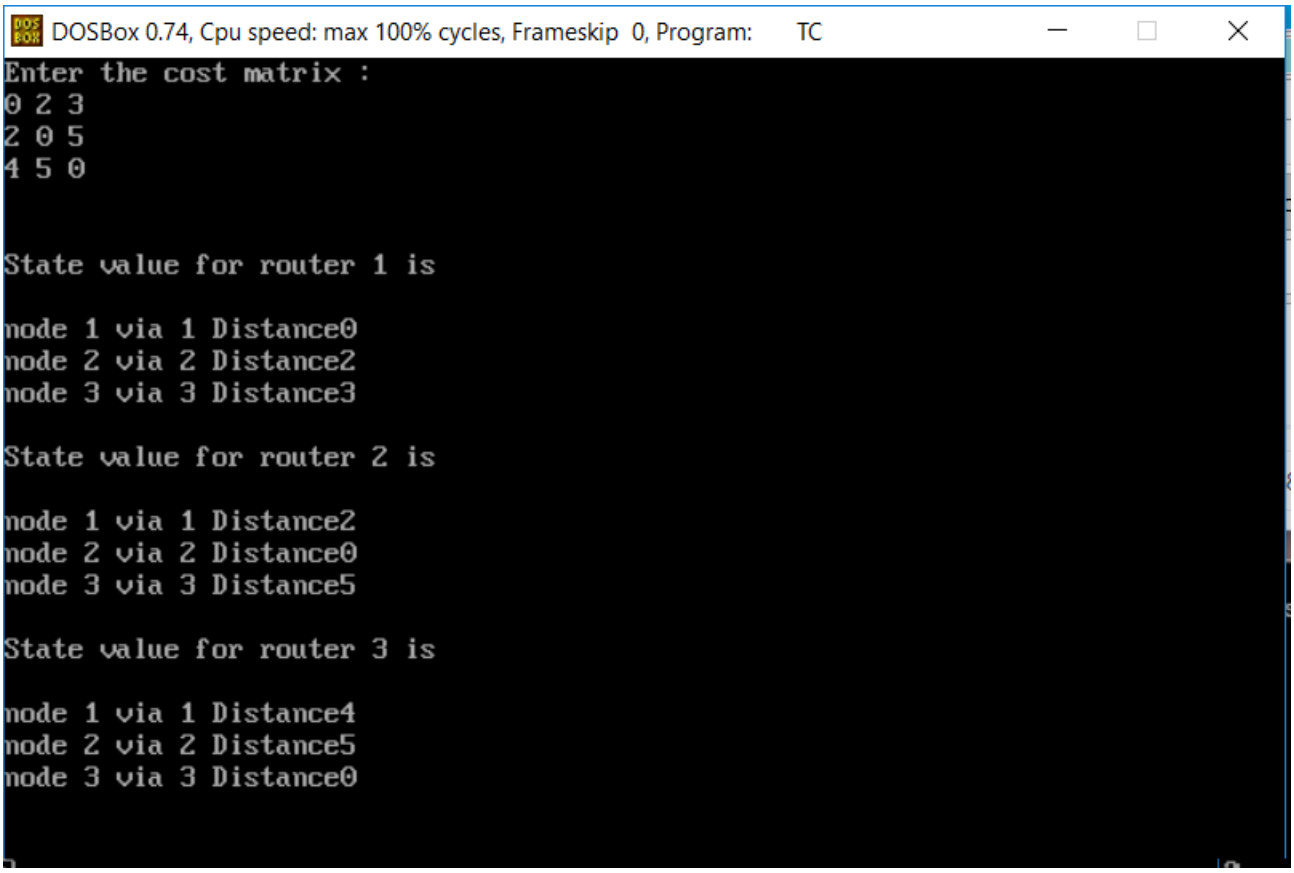
OUTPUT CONSOLE:





Output (2):

Enter the number of nodes: 3


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the cost matrix :
0 2 3
2 0 5
4 5 0
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
node 3 via 3 Distance3
State value for router 2 is
node 1 via 1 Distance2
node 2 via 2 Distance0
node 3 via 3 Distance5
State value for router 3 is
node 1 via 1 Distance4
node 2 via 2 Distance5
node 3 via 3 Distance0

Experiment No: 5

NAME OF THE EXPERIMENT: Broadcast Tree

AIM: Write a C program Implement Broadcast Tree for a given subnet hosts.

SOURCE CODE:

```
#include<stdio.h>
int a[10][10],n;
main()
{
int i,j,root;
clrscr();
printf("Enter no.of nodes:");
scanf("%d",&n);
printf("Enter adjacent matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
printf("Enter connecting of %d>%d::",i,j);
scanf("%d",&a[i][j]);
}
printf("Enter root node:");
scanf("%d",&root);
adj(root);
}
adj(int k)
{
int i,j;
```

```
printf("Adjacent node of root node::\n");
printf("%d\n",k);
for(j=1;j<=n;j++)
{
if(a[k][j]==1 || a[j][k]==1)
printf("%d\t",j);
}
printf("\n");
for(i=1;i<=n;i++)
{
if((a[k][j]==0) && (a[i][k]==0) && (i!=k))
printf("%d",i);
}}
```

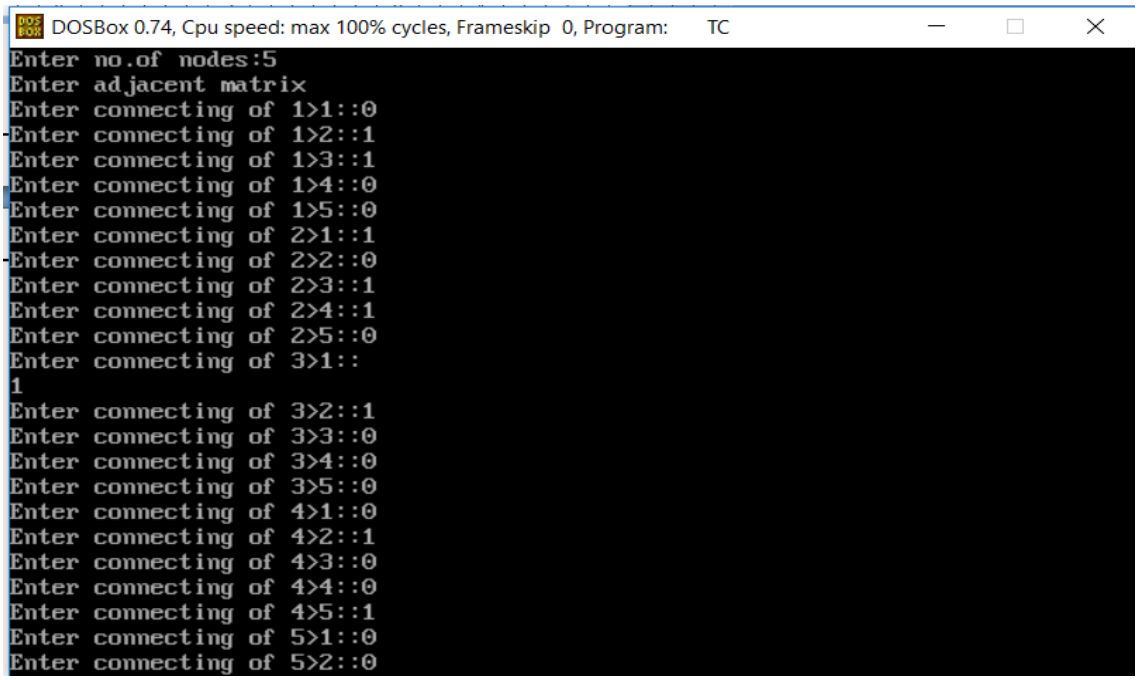
OUTPUT:

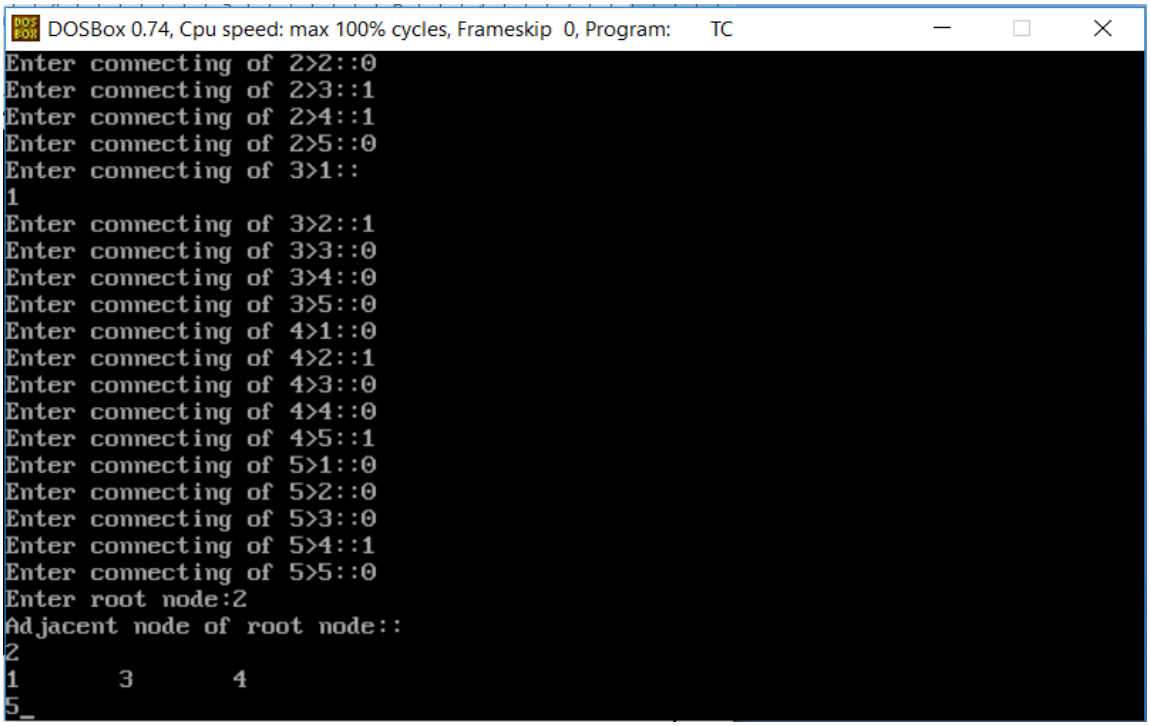
```
Enter no.of nodes:5
Enter adjacent matrix
Enter connecting of 1–>1::0
Enter connecting of 1–>2::1
Enter connecting of 1–>3::1
Enter connecting of 1–>4::0
Enter connecting of 1–>5::0
Enter connecting of 2–>1::1
Enter connecting of 2–>2::0
Enter connecting of 2–>3::1
Enter connecting of 2–>4::1
Enter connecting of 2–>5::0
Enter connecting of 3–>1::1
Enter connecting of 3–>2::1
Enter connecting of 3–>3::0
Enter connecting of 3–>4::0
Enter connecting of 3–>5::0
Enter connecting of 4–>1::0
Enter connecting of 4–>2::1
Enter connecting of 4–>3::0
```

Enter connecting of 4–>4::0
Enter connecting of 4–>5::1
Enter connecting of 5–>1::0
Enter connecting of 5–>2::0

Enter connecting of 5–>3::0
Enter connecting of 5–>4::1
Enter connecting of 5–>5::0
Enter root node:2
Adjacent node of root node::
2
1 3 4
5

OUTPUT CONSOLE:




DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter connecting of 2>2::0
Enter connecting of 2>3::1
Enter connecting of 2>4::1
Enter connecting of 2>5::0
Enter connecting of 3>1::
1
Enter connecting of 3>2::1
Enter connecting of 3>3::0
Enter connecting of 3>4::0
Enter connecting of 3>5::0
Enter connecting of 4>1::0
Enter connecting of 4>2::1
Enter connecting of 4>3::0
Enter connecting of 4>4::0
Enter connecting of 4>5::1
Enter connecting of 5>1::0
Enter connecting of 5>2::0
Enter connecting of 5>3::0
Enter connecting of 5>4::1
Enter connecting of 5>5::0
Enter root node:2
Adjacent node of root node::
2
1 3 4
5_

Experiment No: 6

NAME OF THE EXPERIMENT: Encrypting DES

AIM: Write a C program to implement that to Take a 64 bit playing text and encrypt the same using DES algorithm.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int p10[]={3,5,2,7,4,10,1,9,8,6},
p8[]={6,3,7,4,8,5,10,9},
p4[]={2,4,3,1};
int ip[]={2,6,3,1,4,8,5,7},
ipinv[]={4,1,3,5,7,2,8,6},
ep[]={4,1,2,3,2,3,4,1};
int s0[][4]={{1,0,3,2,},{3,2,1,0},{0,2,1,3,},{3,1,3,2}};
int s1[][4]={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
void permute(char op[],char ip[],int p[], int n){
int i;
for(i=0;i<n;i++)
op[i]=ip[p[i]-1];
op[i]='\0';
}
void circularls(char pr[],int n){
int i;
char ch=pr[0];
for(i=0;i<n-1;i++)
pr[i]=pr[i+1];
pr[i]=ch;
}
void keygen(char k1[],char k2[],char key[]){
char keytemp[11];
```

```
permute(keytemp,key,p10,10);
circularls(keytemp,5);
circularls(keytemp+5,5);
permute(k1,keytemp,p8,8);
circularls(keytemp,5);
circularls(keytemp,5);
circularls(keytemp+5,5);
circularls(keytemp+5,5);
permute(k2,keytemp,p8,8);
}
void xor(char op[],char ip[]){
int i;
for(i=0;i<strlen(op)&&i<strlen(ip);i++)
op[i]=(op[i]-'0')^(ip[i]-'0')+'0';
}
void sbox(char op[],char ip[],int s[][4]) {
int value;
value=s[(ip[0]-'0')*2+(ip[3]-'0')][(ip[1]-'0')*2+(ip[2]-'0')];
op[0]=value/2+'0';
op[1]=value%2+'0';
op[2]='\0';
}
void fk(char op[],char ip[],char k[])
{
char l[5],r[5],tmp[9],tmp1[9],tmp2[9];
strncpy(l,ip,4);
l[4]='\0';
strncpy(r,ip+4,4);
r[4]='\0';
permute(tmp,r,ep,8);
xor(tmp,k);
```

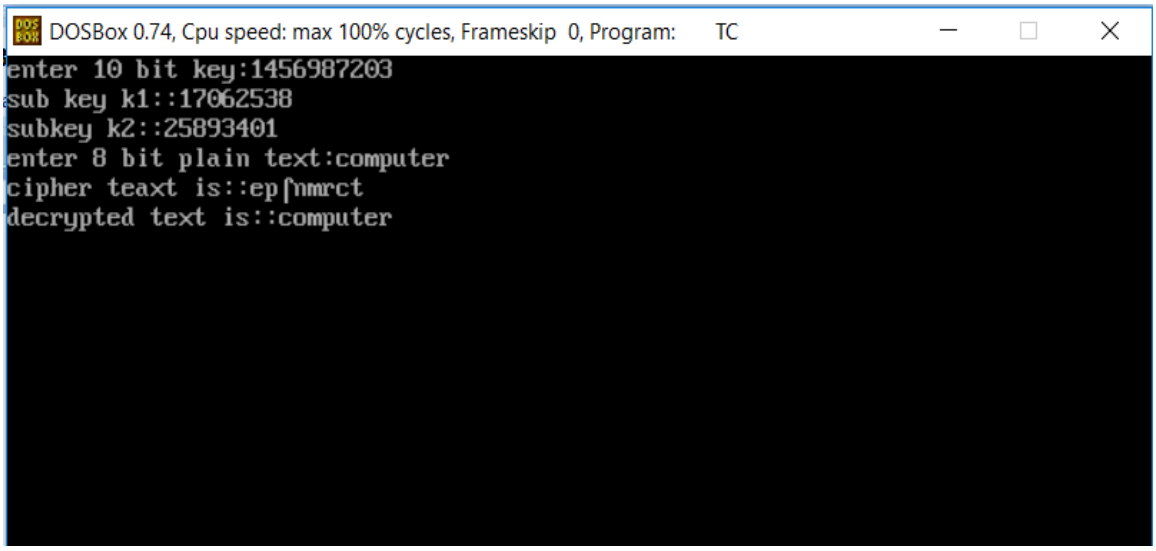
```
sbox(tmp1,tmp,s0);
sbox(tmp2,tmp+4,s1);
strcat(tmp1,tmp2);
permute(tmp,tmp1,p4,4);
xor(tmp,l);
strcat(tmp,r);
strcpy(op,tmp);
}
void sw(char pr[]) {
char tmp[9];
strncpy(tmp,pr+4,4);
strncpy(tmp+4,pr,4);
tmp[8]='\0';
strcpy(pr,tmp);
}
void main()
{
char key[11],k1[9],k2[9],plain[9],cipher[9],tmp[9];
clrscr();
printf("enter 10 bit key:");
gets(key);
if(strlen(key)!=10) printf("invalid key length !!");
else
{
keygen(k1,k2,key);
printf("sub key k1::");
puts(k1);
printf("subkey k2::");
puts(k2);
printf("enter 8 bit plain text:");
gets(plain);
```

```
if(strlen(plain)!=8) printf("invalid length plain text !!");
permute(tmp,plain,ip,8);
fk(cipher,tmp,k1);
sw(cipher);
fk(tmp,cipher,k2);
permute(cipher,tmp,ipinv,8);
printf("cipher teaxt is::");
puts(cipher);
/* decryption process*/
permute(tmp,cipher,ip,8);
fk(plain,tmp,k2);
sw(plain);
fk(tmp,plain,k1);
permute(plain,tmp,ipinv,8);
printf("decrypted text is::");
puts(plain);
}
getch();
}
```

OUTPUT:

```
Enter 10 bit key:1456987203
Sub key k1::17062538
Sub key k2::25893401
Enter 8 bit plain text: computer
Cipher text is::epfnmrct
Decrypted text is::computer
```

OUTPUT CONSOLE:



Experiment No: 7

NAME OD THE EXPERIMENT: Break DES

AIM: Write a c program to implement to break the above DES coding.

SOURCE CODE:

```
 #include<stdio.h>
#include<conio.h>
#include<string.h>
int p10[]={3,5,2,7,4,10,1,9,8,6},
p8[]={6,3,7,4,8,5,10,9},
p4[]={2,4,3,1};
int ip[]={2,6,3,1,4,8,5,7},
ipinv[]={4,1,3,5,7,2,8,6},
ep[]={4,1,2,3,2,3,4,1};
int s0[][4]={{1,0,3,2,},{3,2,1,0},{0,2,1,3,},{3,1,3,2}};
int s1[][4]={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
void permute(char op[],char ip[],int p[], int n)
{
int i;
for(i=0;i<n;i++)
op[i]=ip[p[i]-1];
op[i]='\0';
}
void circularls(char pr[],int n)
{
int i;
char ch=pr[0];
for(i=0;i<n-1;i++)
pr[i]=pr[i+1];
pr[i]=ch;
}
```

```
void keygen(char k1[],char k2[],char key[])
{

char keytemp[11];
permute(keytemp,key,p10,10);
circularls(keytemp,5);
circularls(keytemp+5,5);
permute(k1,keytemp,p8,8);
circularls(keytemp,5);
circularls(keytemp,5);
circularls(keytemp+5,5);
circularls(keytemp+5,5);
permute(k2,keytemp,p8,8);
}
void xor(char op[],char ip[])
{
int i;
for(i=0;i<strlen(op)&&i<strlen(ip);i++)
op[i]=(op[i]-'0')^(ip[i]-'0')+'0';
}
void sbox(char op[],char ip[],int s[][4])
{
int value;
value=s[(ip[0]-'0')*2+(ip[3]-'0')][(ip[1]-'0')*2+(ip[2]-'0')];
op[0]=value/2+'0';
op[1]=value%2+'0';
op[2]='\0';
}
void fk(char op[],char ip[],char k[])
{
char l[5],r[5],tmp[9],tmp1[9],tmp2[9];
```

```
strncpy(l,ip,4);
l[4]='\0';
strncpy(r,ip+4,4);
r[4]='\0';
permute(tmp,r,ep,8);
xor(tmp,k);
sbox(tmp1,tmp,s0);
sbox(tmp2,tmp+4,s1);
strcat(tmp1,tmp2);
permute(tmp,tmp1,p4,4);
xor(tmp,l);
strcat(tmp,r);
strcpy(op,tmp);
}
void sw(char pr[])
{
char tmp[9];
strncpy(tmp,pr+4,4);
strncpy(tmp+4,pr,4);
tmp[8]='\0';
strcpy(pr,tmp);
}
void main()
{
char key[11],k1[9],k2[9],plain[9],cipher[9],tmp[9];
clrscr();
printf("enter 10 bit key:");
gets(key);
if(strlen(key)!=10) printf("invalid key length !!");
else
{
```

```
keygen(k1,k2,key);
printf("sub key k1::");
puts(k1);
printf("subkey k2::");
puts(k2);
printf("enter 8 bit plain text:");
gets(plain);
if(strlen(plain)!=8) printf("invalid length plain text !!");
permute(tmp,plain,ip,8);
fk(cipher,tmp,k1);
sw(cipher);
fk(tmp,cipher,k2);
permute(cipher,tmp,ipinv,8);
printf("cipher teaxt is::");
puts(cipher);
/* decryption process*/
permute(tmp,cipher,ip,8);
fk(plain,tmp,k2);
sw(plain);
fk(tmp,plain,k1);
permute(plain,tmp,ipinv,8);
printf("decrypted text is::");
puts(plain);
}
getch();
}
```

OUTPUT:

Enter 10 bit key: 1234567892

Sub key k1::17948326

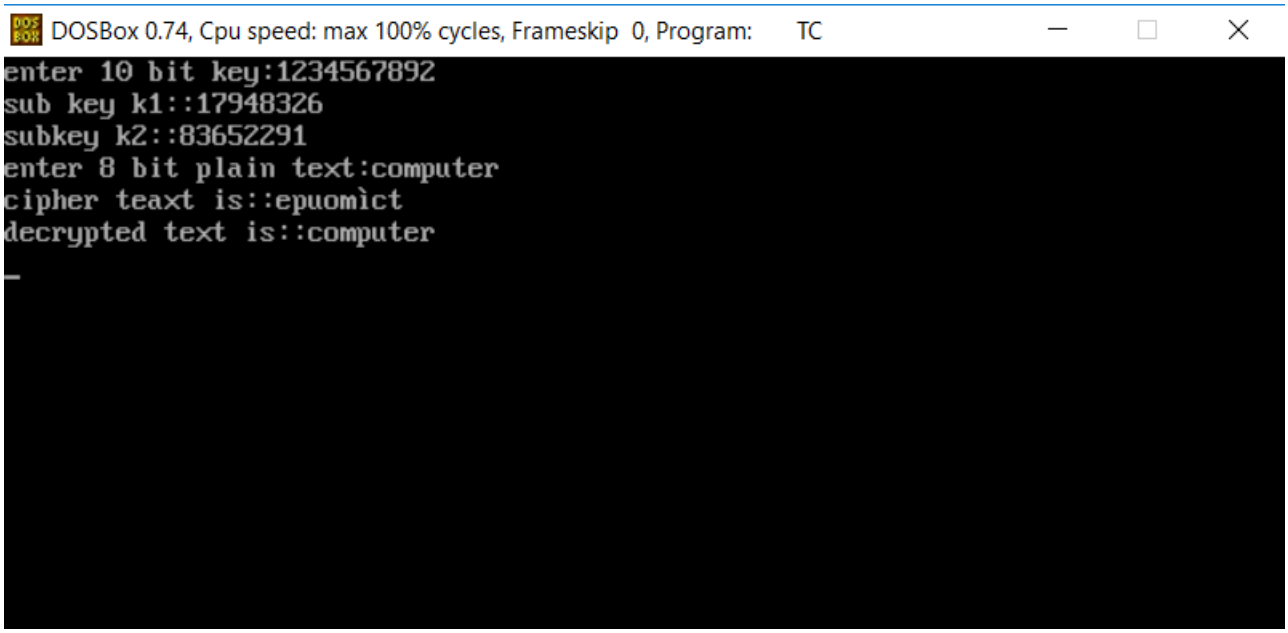
Sub key k2::83652291

Enter 8 bit plain text: computer

Cipher text is:: epuomict

Decrypted text is :: computer

OUTPUT CONSOLE:



Experiment No: 8

NAME OF THE EXPERIMENT: Encryption and Decryption of RSA algorithm

AIM: Using RSA algorithm encrypts a text data and Decrypt the same.

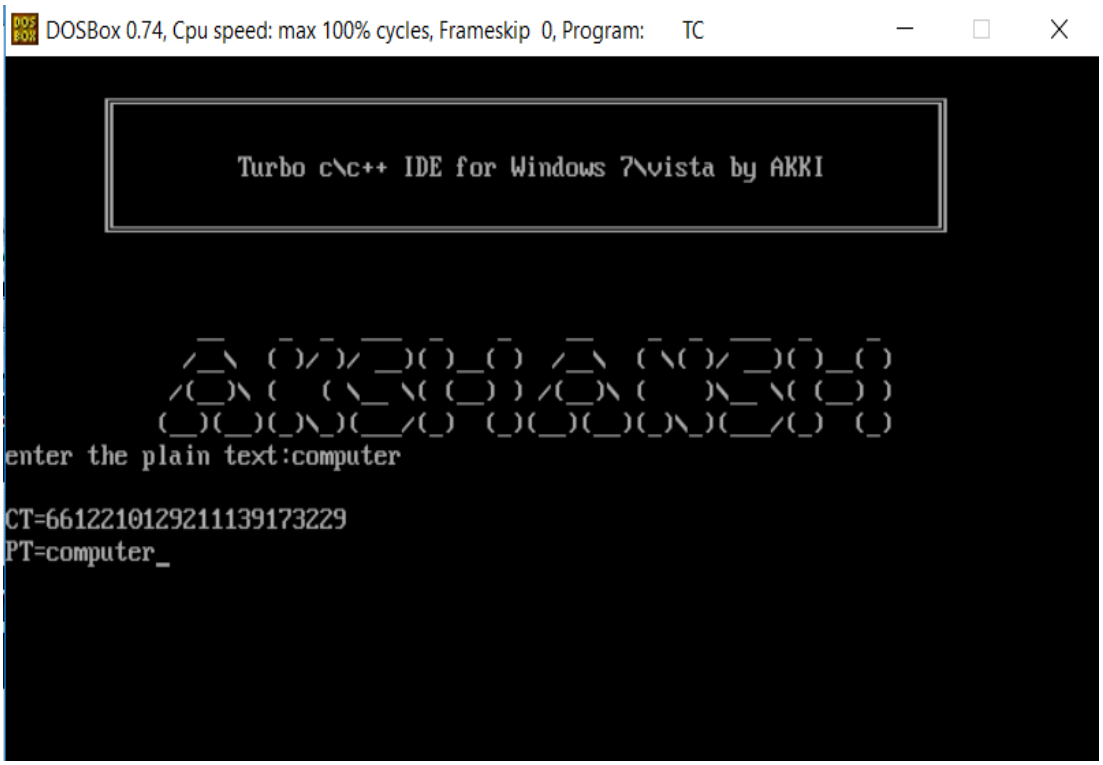
SOUECE CODE:

```
 #include<stdio.h>
#include<string.h>
#include<conio.h>
#include<math.h>
int mult(unsigned int x,unsigned int y,unsigned int n)
{
unsigned long int k=1;
int j;
for(j=1;j<=y;j++)
k=(k*x)%n;
return(unsigned int) k;
}
void main()
{
char msg[100];
unsigned int pt[100],ct[100],n,d,e,p,q,i;
printf("enter the plain text:");
gets(msg);
//strcpy(pt,msg);
for(i=0;i<strlen(msg);i++)
pt[i]=msg[i];
n=253;d=17;e=13;
printf("\nCT=");
for(i=0;i<strlen(msg);i++)
ct[i]=mult(pt[i],e,n);
for(i=0;i<strlen(msg);i++)
printf("%d",ct[i]);
printf("\nPT=");
for(i=0;i<strlen(msg);i++)
printf("%c",pt[i]);
for(i=0;i<strlen(msg);i++)
pt[i]=mult(ct[i],d,n);
}
```

OUTPUT:

Enter the plain text: computer
CT=66 119 10 129211 139 173 229
PT=computer

OUTPUT CONSOLE:

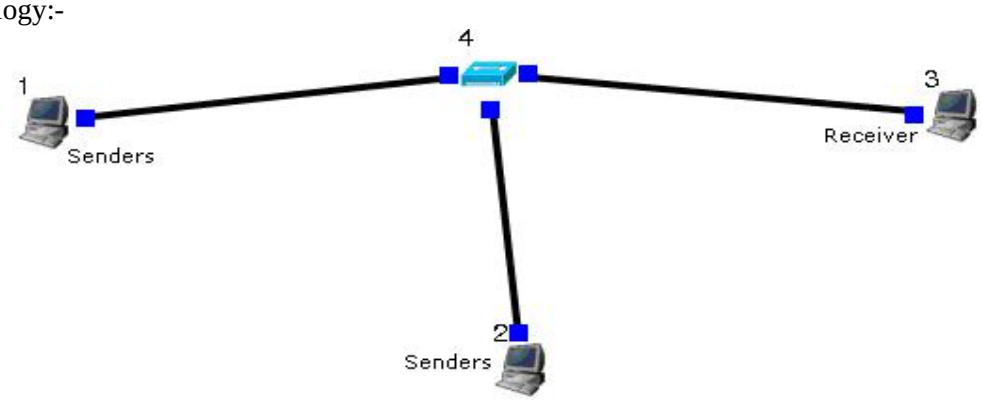


ADDITIONAL EXPERIMENTS:

1. **Name of the Program:** Simulate to Find the Number of Packets Dropped by TCP/UDP

   AIM: Simulate a four-node point-to-point network and connect the link as follows: Apply a TCP agent between n0 to n3 and apply a UDP agent between n1 and n3. Apply relevant applications over TCP and UDP agents changing the parameters and determine the number of packets sent by two agents.

   Topology:-



Sender:-
stcp –p 3000 –l 1024 1.0.1.3
stg –u 1024 1.0.1.3

Receiver:-
rtcp –p 3000 –l 1024
rtg –u 3000

Parameters:-
Throughput of incoming and outgoing Packets

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place a host on the editor. Repeat the above procedure and place two other hosts “HOST2” and “HOST3” on the editor.

1. Select/click the HUB (or SWITCH) icon on the toolbar and click the left mouse button on the editor, to place a HUB (or SWITCH) on the editor.
3. Click on the LINK icon on the toolbar and connect HOST1 to HUB, HOST2 to HUB and HUB to HOST3

4. Click on the “E” icon on the toolbar to save the current topology e.g: file2.tpl (Look for the ******.tpl extension.)
NOTE: Changes cannot / (should not) be done after selecting the “E” icon.

Step2: Configuration
1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the

following command in the command textbox. stcp –p 21 –l 1024 1.0.1.3

3. Click OK button on the command window to exit

4. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

5. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

6. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

7. Double click the left mouse button while cursor is on HOST2 to open the HOST window.

8. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

stg –u 1024 100 1.0.1.3

9. Click OK button on the command window to exit

10. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

11. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

12. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

13. Double click the left mouse button while cursor is on HOST3 to open the HOST window.

14. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.
rtcp –p 21 –l 1024

15. Click OK button on the command window to exit.

16. Also add the following command on HOST3
rtg –u –w log1

17. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

18. Select LOG STATISTICS and select checkbox for input and output throughput in the MAC window

19. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

Step3: Simulate

i. Click “R” icon on the tool bar

ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.

iii. To start playback select “►” icon located at the bottom right corner of the editor.

iv. To view results, Open up new TERMINAL window, move to file2.results folder and open input and output throughput log files in separate TERMINAL window.

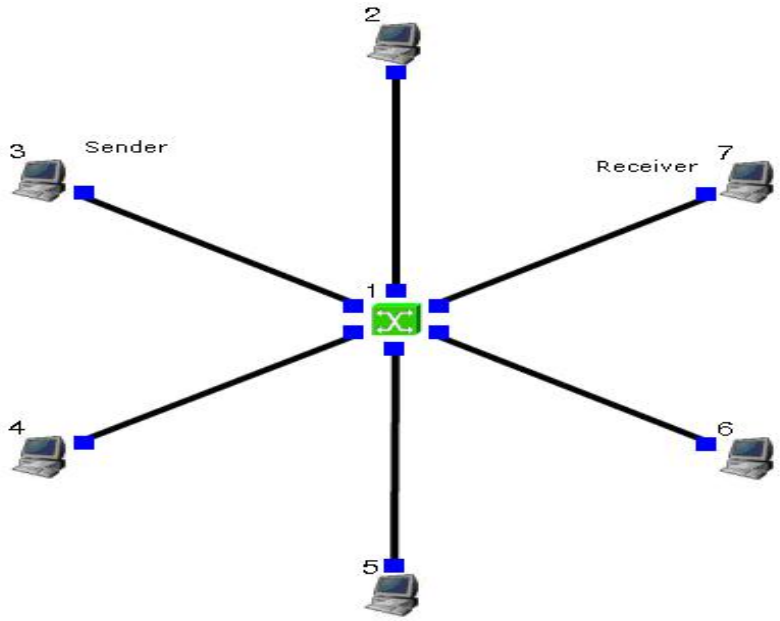
Caution: file2 is the hypothetical name given to this simulation.

(Refer Step 1.4)

2. **Name of the Program:** Simulate to Find the Number of Packets Dropped due to Congestion.

AIM: Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

Topology:-



Sender:-
stcp –p 2000 –l 1024 1.0.1.4

Receiver:-
rtcp –p 2000 –l 1024

Command Console:-

Goto tools-> simulation time and change Simulation time to 100. During run mode, double click host 2 and then click command console. And execute the following command.

ping 1.0.1.4

Parameters:-
Drop Packets and Collision Packets.
Step1: Drawing topology
1. Select/click the SUBNET icon on the toolbar and click the left mouse button on the editor, to place a SUBNET on the editor.

2. A pop up window appears requesting the number of nodes and radius for the subnet

Set number of nodes=6;
Set radius of subnet >150

3. Click on the “E” icon on the toolbar to save the current topology e.g: file4.tpl (Look for the ******.tpl extension.)
NOTE: Changes cannot / (should not) be done after selecting

Step2 : Configuration

4. Double click the left mouse button while cursor is on a HOST to open the HOST window.

5. Click NODE EDITOR Button on the HOST window and select the INTERFACE tab (1st tab) from the modal window that pops up.

6. Determine the IP address of the selected host.

7. Click OK button on the INTERFACE window to exit and once again click on the OK button on the HOST window to exit.

8. Repeat the above step for 2 other HOSTS

9. Also click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

10. Select LOG STATISTICS and select checkbox for drop and collision log statistics in the MAC window

11. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

12. Repeat steps 6 to 9 for the other hosts selected at step 5.

13. Select G_Setting from the menu bar and select Simulation from the drop down list
Set simulation time>600sec

Step3: Simulate
i. Click “R” icon on the tool bar

ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.

iii. During simulation, open a new terminal window.

iv. Type ping IP address of a host in the subnet at the command prompt.

v. To view results, Open up new TERMINAL window, move to file4.results folder and open drop and collision log files in separate TERMINAL window.
Caution: file4 is the hypothetical name given to this simulation.

(Refer Step 1.3)