

Laboratory Manual

of

Data Communication and computer Networks



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG COLLEGE OF ENGINEERING

Aushapur (V), Ghatkesar (M), Medchal Dist.

EXPERIMENT NO: 1

1(a)

NAME OF THE EXPERIMENT: Bit Stuffing.

OBJECTIVE: Implement the data link layer framing method.

RESOURCE: Turbo C

PROGRAM LOGIC:

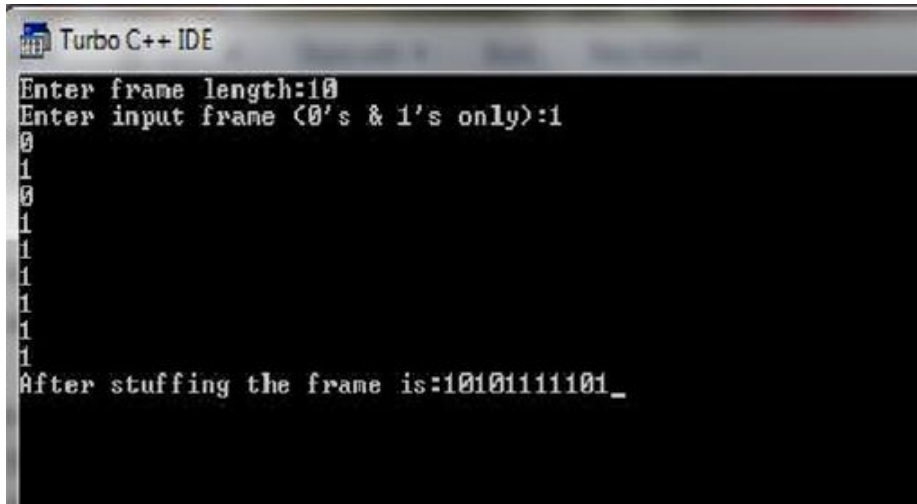
The new technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data.

SOURCE CODE:

```
// BIT Stuffing program
#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);
printf("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:



```
Turbo C++ IDE  
Enter frame length:10  
Enter input frame (0's & 1's only):1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
After stuffing the frame is:10101111101_
```

Viva questions:

1. What is Stuffing?
2. What is use of Stuffing?
3. With bit stuffing the boundary between two frames can be unambiguously recognize by?
4.is a analogous to character stuffing?
5. The senders data link layer encounters.....no of 1's consecutively

EXPERIMENT NO: 1

1(b)

NAME OF THE EXPERIMENT: Character Stuffing.

OBJECTIVE: Implement the data link layer framing methods.

RESOURCE: Turbo C

PROGRAM LOGIC:

The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever loses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

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';
```

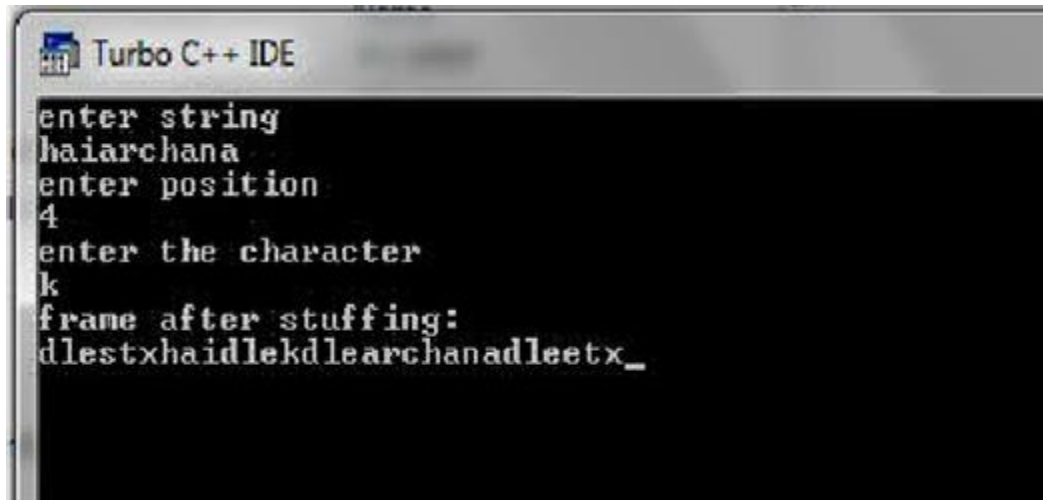
```
b[5]='x';
```

```
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("\nframe after stuffing:\n");
printf("%s",b);
getch();
}
```

OUTPUT:



```
Turbo C++ IDE
enter string
haiarchana
enter position
4
enter the character
k
frame after stuffing:
dlestxhaidlekdlearchanadleetx_
```

Viva Questions:

1. What is Character stuffing?
2. What is the use of character stuffing?
3. -----are the delimiters for the character stuffing?
4. Expand DLE STX?
5. Expand DLE ETX?

EXPERIMENT NO: 2

NAME OF THE EXPERIMENT: Cyclic Redundancy Check.

OBJECTIVE: Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

RESOURCE: Turbo C

PROGRAM LOGIC:

CRC method can detect a single burst of length n , since only one bit per column will be changed, a burst of length $n+1$ will pass undetected, if the first bit is inverted, the last bit is inverted and all other bits are correct. If the block is badly garbled by a long burst or by multiple shorter burst, the probability that any of the n columns will have the correct parity that is 0.5. so the probability of a bad block being expected when it should not be 2^{-n} . This scheme sometimes known as Cyclic Redundancy Code

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
//PROGRAM FOR CYCLIC REDUNDENCY CHECK
```

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
int gen[4],genl,frl,rem[4];
```

```
void main()
```

```
{
```

```
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
```

```
clrscr();
```

```
frl=8;
```

```
genl=4;
```

```
printf("enter frame:");
```

```
for(i=0;i<frl;i++)
```

```
{
```

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

```
dupfr[i]=fr[i];
```

```
}
```

```
printf("enter generator:");
```

```
for(i=0;i<genl;i++)
```

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

```
tlen=frl+genl-1;
```

```
for(i=frl;i<tlen;i++)
```

```
{
```

```
dupfr[i]=0;
```

```
}
```

```
remainder(dupfr);
```

```
for(i=0;i<frl;i++)
```

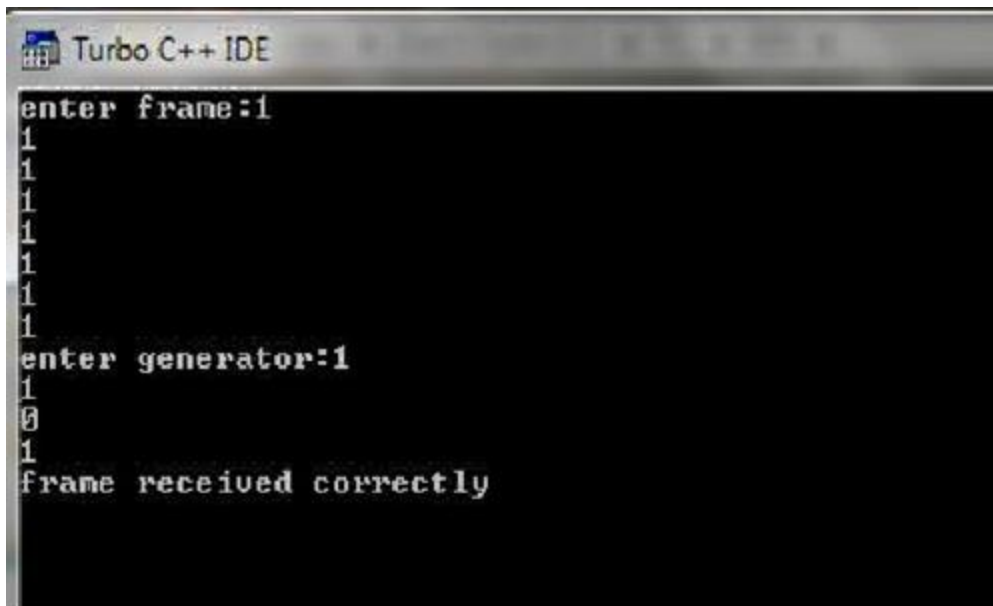
```
{
```

```
recfr[i]=fr[i];
```

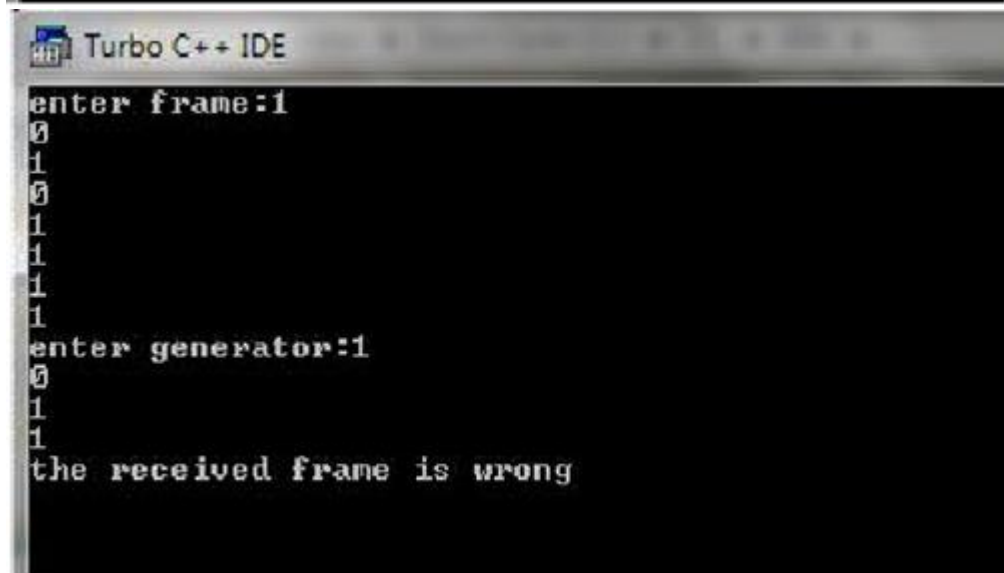
```
}
```

```
for(i=frl,j=1;j<genl;i++,j++)
{
    recfr[i]=rem[j];
}
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
    if(rem[i]!=0)
        flag++;
}
if(flag==0)
{
    printf("frame received correctly");
}
Else
{
    printf("the received frame is wrong");
}
getch();
}
remainder(int fr[])
{
    int k,k1,i,j;
    for(k=0;k<frl;k++)
    {
        if(fr[k]==1)
        {
            k1=k;
            for(i=0,j=k;i<genl;i++,j++)
            {
                rem[i]=fr[j]^gen[i];
            }
            for(i=0;i<genl;i++)
            {
                fr[k1]=rem[i];
                k1++;
            }
        }
    }
}
```


OUTPUT:



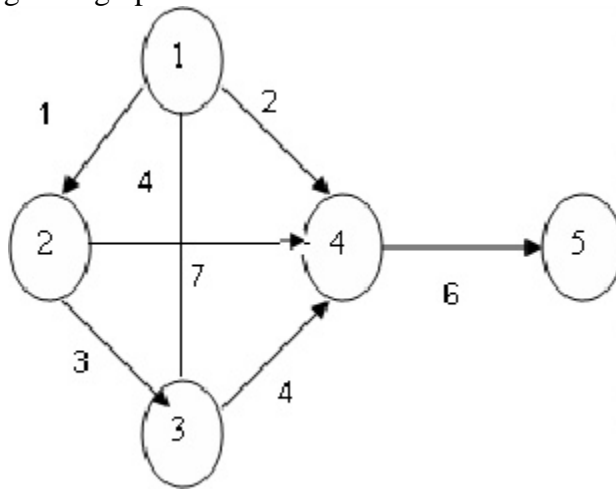
```
Turbo C++ IDE
enter frame:1
1
1
1
1
1
1
1
1
1
1
enter generator:1
1
0
1
frame received correctly
```



```
Turbo C++ IDE
enter frame:1
0
1
0
1
1
1
1
1
1
enter generator:1
0
1
1
the received frame is wrong
```

Viva Questions:

1. What is CRC?
2. What is the use of the CRC?
3. Name the CRC standards?
4. Define Checksum?
5. Define generator polynomial?

EXPERIMENT NO: 3**NAME OF THE EXPERIMENT:** Shortest Path.**OBJECTIVE:** Implement Dijkstra's algorithm to compute the Shortest path thru a given graph.**RESOURCE:** Turbo C

Program Logic: Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.


SOURCE CODE:

```
//.PROGRAM FOR FINDING SHORTEST PATH FOR A GIVEN GRAPH//
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
clrscr();
printf("enter the cost matrix\n");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d",&a[i][j]);
printf("enter the paths\n");
scanf("%d",&p);
printf("enter possible paths\n");
for(i=1;i<=p;i++)
for(j=1;j<=5;j++)
scanf("%d",&path[i][j]);
```

```
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=5;j++)
{
edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else
stp=edp;
}
}min=t[st];
index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];
index=i;
}
}
printf("minimum cost %d",min);
printf("\n minimum cost path ");
for(i=1;i<=5;i++)
{
printf("--> %d",path[index][i]);
if(path[index][i]==ed)
break;
}
getch();
}
```

Output:



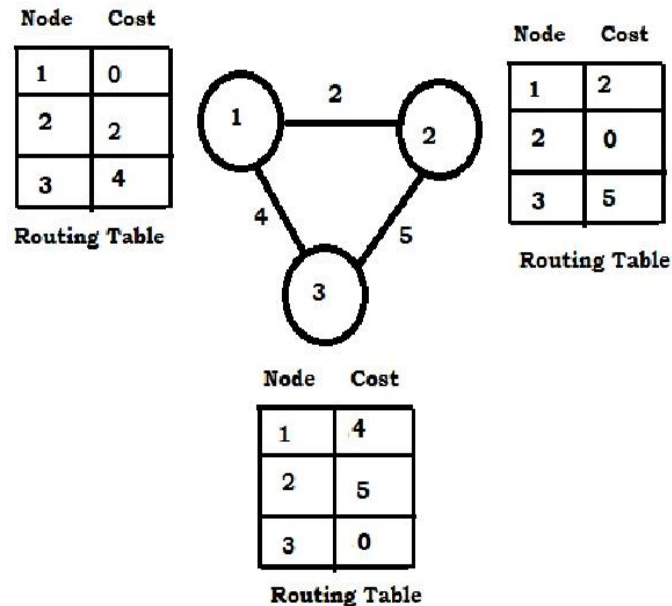
```

Turbo C++ IDE
enter the cost matrix
0 1 4 2 0
1 0 3 7 0
4 3 0 5 0
2 7 5 0 6
0 0 0 6 0
enter the paths
4
enter possible paths
1 2 3 4 5
1 2 4 5 0
1 3 4 5 0
1 4 5 0 0
minimum cost 8
minimum cost path --> 1--> 4--> 5_

```

Viva questions:

1. Define Dijkstra's algorithm?
2. What is the use of Dijkstra's algorithm?
3. What is path?
4. What is minimum cost path?
5. How to find shortest path using Dijkstra's algorithm?

EXPERIMENT NO: 4**NAME OF THE EXPERIMENT:** Distance Vector routing.**OBJECTIVE:** Obtain Routing table at each node using distance vector routing algorithm for a given subnet.**RESOURCE:** Turbo C**PROGRAM LOGIC:**

Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination. Distance vector routing algorithms are not preferable for complex networks and take longer to converge.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.**SOURCE CODE:**

```
#include<stdio.h>

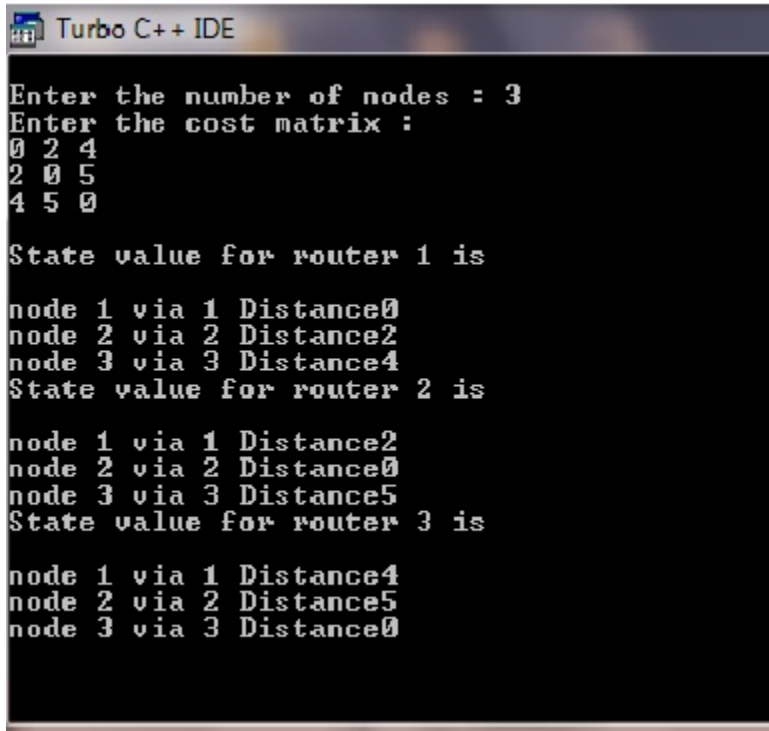
#include<conio.h>

struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];

int main()
```

```
{
int dmat[20][20];
int n,i,j,k,count=0;
clrscr();
printf("\nEnter the number of nodes : ");
scanf("%d",&n);printf("Enter 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("\nState value for router %d is \n",i+1);
for(j=0;j<n;j++)
{
printf("\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n");
}
```

Output:



```
Turbo C++ IDE

Enter the number of nodes : 3
Enter the cost matrix :
0 2 4
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 Distance4
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
```

Viva Questions:

1. What is routing?
2. What is best algorithm among all routing algorithms?
3. What is static routing?
4. Differences between static and dynamic?
5. What is optimality principle?

EXPERIMENT NO: 5

NAME OF THE EXPERIMENT: Broadcast Tree.

OBJECTIVE: Implement broadcast tree for a given subnet of hosts

RESOURCE: Turbo C

PROGRAM LOGIC:

This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers the algorithm just finds the broadcast between them on the graph.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
// Write a 'c' program for Broadcast tree from subnet of host
#include<stdio.h>
#include<conio.h>
int p,q,u,v,n;
int min=99,mincost=0;
int t[50][2],i,j;
int parent[50],edge[50][50];
main()
{
clrscr();
printf("\n Enter the number of nodes");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("%c\t",65+i);
parent[i]=-1;
}
printf("\n");
for(i=0;i<n;i++)
{
printf("%c",65+i);
for(j=0;j<n;j++)
scanf("%d",&edge[i][j]);
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
if(edge[i][j]!=99)
if(min>edge[i][j])
{
min=edge[i][j];
u=i;
}
```



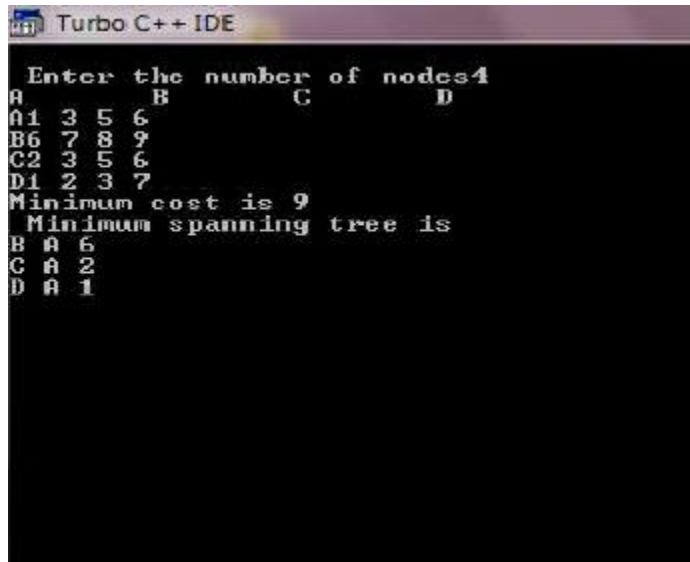
```
v=j;
}
p=find(u);
q=find(v);
if(p!=q)
{
t[i][0]=u;
t[i][1]=v;
mincost=mincost+edge[u][v];
union(p,q);
}

Else
{
t[i][0]=-1;t[i][1]=-1;
}
min=99;
}
printf("Minimum cost is %d\n Minimum spanning tree is\n",mincost);
for(i=0;i<n;i++)
if(t[i][0]!=-1 && t[i][1]!=-1)
{
printf("%c %c %d", 65+t[i][0],65+t[i][1],edge[t[i][0]][t[i][1]]);printf("\n");
}
getch();
}
union(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)

i=parent[i];

return i;
}
```

Output:



```
Turbo C++ IDE
Enter the number of nodes: 4
A      B      C      D
A 1 3 5 6
B 6 7 8 9
C 2 3 5 6
D 1 2 3 7
Minimum cost is 9
Minimum spanning tree is
B A 6
C A 2
D A 1
```

Viva questions:

1. What is spanning tree?
2. What is broad cast tree?
3. What are the advantages of broadcast tree?
4. What is flooding?
5. What is subnet?

EXPERIMENT NO: 6

NAME OF THE EXPERIMENT: encrypting DES.

OBJECTIVE: Take a 64 bit playing text and encrypt the same using DES algorithm.

RESOURCE: Turbo C

PROGRAM LOGIC:

Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.

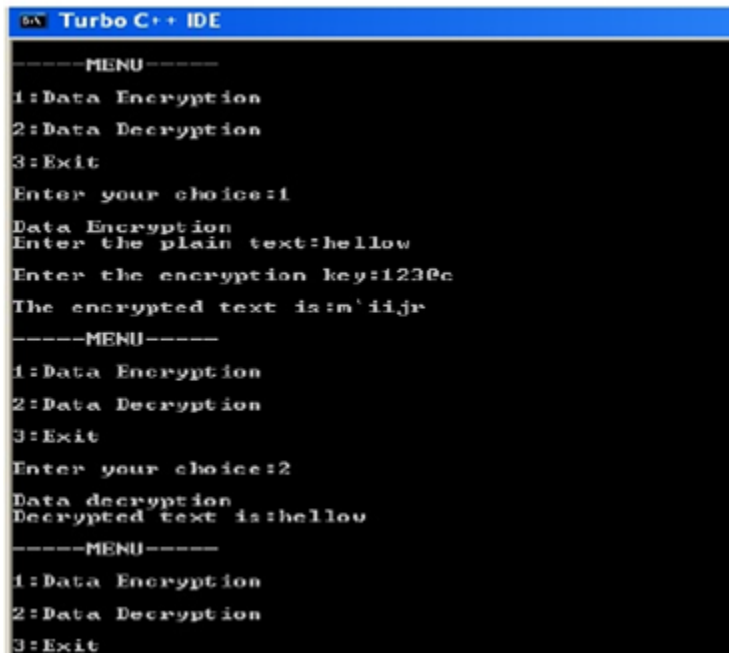
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
int i,ch,lp;char cipher[50],plain[50];
char key[50];
clrscr();
while(1)
{
printf("\n----MENU-----\n");
printf("\n1:Data Encryption\t\n2:Data Decryption\t\n3:Exit");
printf("\n\nEnter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("\nData Encryption");
printf("\nEnter the plain text:");
fflush(stdin);
gets(plain);
printf("\nEnter the encryption key:");
gets(key);
lp=strlen(key);
for(i=0;plain[i]!='\0';i++)
cipher[i]=plain[i]^lp;
cipher[i]='\0';
printf("\nThe encrypted text is:");
puts(cipher);
break;
```

```
case 2:
printf("\nData decryption");
for(i=0;cipher[i]!='\0';i++)
plain[i]=cipher[i]^lp;
printf("\nDecrypted text is:");
puts(plain);
break;
case 3: exit(0);
}
}
getch();
}
```

Output:



```
Turbo C++ IDE
-----MENU-----
1:Data Encryption
2:Data Decryption
3:Exit
Enter your choice:1
Data Encryption
Enter the plain text:hellow
Enter the encryption key:1230e
The encrypted text is:m'ijr
-----MENU-----
1:Data Encryption
2:Data Decryption
3:Exit
Enter your choice:2
Data decryption
Decrypted text is:hellow
-----MENU-----
1:Data Encryption
2:Data Decryption
3:Exit
```

Viva Questions:

1. Expand DES
2. What is cipher text?
3. What is plain text?
4. Define public key?
5. Define encryption?

EXPERIMENT NO: 7

NAME OF THE EXPERIMENT: Decrypting DES.

OBJECTIVE: Write a program to break the above DES coding

RESOURCE: Turbo C

PROGRAM LOGIC:

Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.

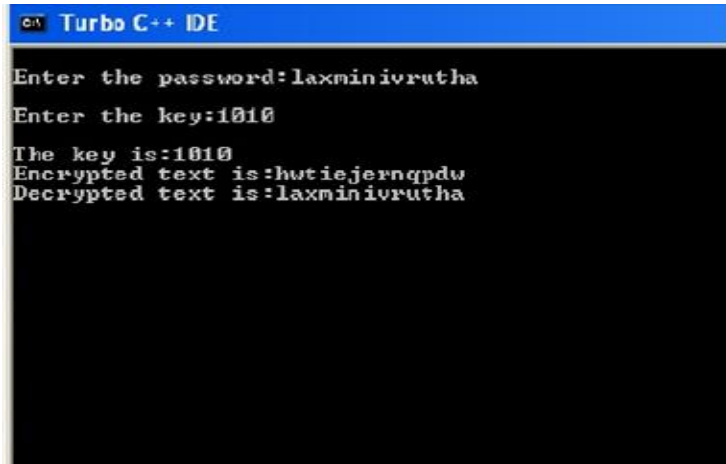
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
/*Write a program to break the above DES coding*/
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
void main()
{
char pwd[20];
char alpha[26]="abcdefghijklmnopqrstuvwxyz";
int num[20],i,n,key;
clrscr();
printf("\nEnter the password:");
scanf("%s",&pwd);
n=strlen(pwd);
for(i=0;i<n;i++)
num[i]=toascii(tolower(pwd[i]))-'a';
printf("\nEnter the key:");
scanf("%d",&key);
for(i=0;i<n;i++)
num[i]=(num[i]+key)%26;
for(i=0;i<n;i++)
pwd[i]=alpha[num[i]];
printf("\nThe key is:%d",key);
printf("\nEncrypted text is:%s",pwd);
for(i=0;i<n;i++)
{
num[i]=(num[i]-key)%26;
if(num[i]<0)
num[i]=26+num[i];
pwd[i]=alpha[num[i]];
}
```

```
printf("\nDecrypted text is:%s",pwd);  
getch();  
}
```

Output:

A screenshot of the Turbo C++ IDE window. The title bar is blue and says "Turbo C++ IDE". The main window has a black background with white text. The text displayed is: "Enter the password:laxminivrutha", "Enter the key:1010", "The key is:1010", "Encrypted text is:hwtiejernqpdw", and "Decrypted text is:laxminivrutha".

```
Enter the password:laxminivrutha  
Enter the key:1010  
The key is:1010  
Encrypted text is:hwtiejernqpdw  
Decrypted text is:laxminivrutha
```

Viva Questions:

1. Define decryption
2. What is private key?
3. What is cipher feedback mode?
4. Define product cipher
5. What is DES chaining?

EXPERIMENT NO: 8

NAME OF THE EXPERIMENT: RSA.

OBJECTIVE: Using RSA algorithm encrypt a text data and Decrypt the same.

RESOURCE: Turbo C

PROGRAM LOGIC:

^RSA method is based on some principles from number theory. In encryption process divide the plain text into blocks, so that each plain text message p falls in the interval $0 < p < n$ this can be done by grouping the plain text into blocks of k bits. Where k is the largest integer for which $2^k < n$ is true. The security of this method is based on the difficulty of factoring large numbers. The encryption and decryption functions are inverses

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
/*Using RSA algorithm encrypt a text data and Decrypt the same*/
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<math.h>
#include<string.h>
void main()
{
int a,b,i,j,t,x,n,k=0,flag=0,prime[100];
char m[20],pp[20];float p[20],c[20];
double e,d;
clrscr();
for(i=0;i<50;i++)
{
flag=0;
for(j=2;j<i/2;j++)
if(i%j==0)
{
flag=1;break;
}
if(flag==0)
prime[k++]=i;
}
a=prime[k-1];
b=prime[k-2];
n=a*b;
t=(a-1)*(b-1);
e=(double)prime[2];
d=1/(float)e;
printf("\nKey of encryption is:%lf\n",d);
printf("\nEnter plain the text:");
scanf("%s",&m);
```

```

x=strlen(m);
printf("\nDecryption status From Source to Destination:\n");
printf("\nSource\t->-----<-destination\n");
printf("\nChar\tnumeric\tcipher\t\tnumeric\t\tchar \n");
printf("\n*****\n");
printf("\n");
for(i=0;i<x;i++)
{
printf("%c",m[i]);
printf("\t%d",m[i]-97);
c[i]=pow(m[i]-97,(float)e);
c[i]=fmod(c[i],(float)n);
printf("\t%f",c[i]);
p[i]=pow(c[i],(float)d);
p[i]=fmod(p[i],(float)n);
printf("\t%f",p[i]);pp[i]=p[i]+97;
printf("\t%c\n",pp[i]);
printf("\n*****\n");
printf("\n");
}
getch();
}

```

Output:

```

Turbo C++ IDE
Key of encryption is:0.500000
Enter plain the text:cseab
Decryption status From Source to Destination:
Source  ->-----<-destination
Char    numeric cipher          numeric      char
*****
c        2        4.000000        2.000000        c
*****
s        18       324.000000       18.000000        s
*****
e        4        16.000000        4.000000        e
*****
a        0         0.000000         0.000000        a
*****
b        1         1.000000         1.000000        b
*****

```


Viva Questions:

1. Expand RSA
2. What is encryption and decryption in RSA?
3. To encrypt a message P, Compute $c = \text{-----}$?
4. To compute c compute $P = \text{-----}$?
5. Define cryptography.

ADDITIONAL PROGRAMS

EXPERIMENT NO: 9

NAME OF THE EXPERIMENT: FTP.

OBJECTIVE: Write a c program for FTP protocol.

RESOURCE: Turbo C

PROGRAM LOGIC: The File Transfer **Protocol (FTP)** is a standard network **protocol** used for the transfer of computer files between a client and server on a computer network. **FTP** is built on client-server model architecture and uses separate control and data connections between the client and the server.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

Source Program:

//Server Program:

```
#include<stdio.h>
```

```
#include<sys/socket.h>
```

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

```
#include<stdlib.h>
```

```
#include<unistd.h>
```

```
#include<string.h>
```

```
#include<netdb.h>
```

```
#include<arpa/inet.h>
```

```
#include<sys/types.h>
```

```
#include<time.h>
```

```
#define MAX 100;
```

```
int main ()
```

```
{
```

```
int listenfd,connfd,n,nl;
```

```
struct sockaddr_in serv;
```

```
char str1 [100],str2[100],fname[20],fname1[20],s[2],
```

```
int port=9999;
```

```
FILE*f1*f2;

listenfd=socket(AF_INET,SOCK_STREAM,0);

bzero(&serv,sizeof(serv);

serv.sin_family=AF_INET;

serv.sin_addr.s_addr=htonl(INADDR_ANY);

serv.sin_port=htons(port);

bind(listenfd,(struct sockaddr*)&serv,sizeof(serv));

"serftp4.cpp"54L, 1518C

listen(listenfd,5);

for(;;)

{

connfd=accept(listenfd,(struct sockaddr*)NULL,NULL);

printf("\nClient requesting");

n=read(connfd,fname,20);

fname(n)='\0';

printf("\n Received:%s\n",fname);

f1=fopen(fname)."r");

strcpy(s,"ab");

write(connfd,s,strlen(s));

n1=read(connfd,fname1,20);

fname1[n1]='\0';

printf("stored in:%s\n"fname 1);

f2=fopen(fname1,"w");

while(!feof(f1))

{

fgets(str1,50f1);

write(connfd,str,1,strlen(str1));

n=read(connfd,str,2,100);
```

```
str2[n]='\0';  
fputs(str2,f2);  
}  
fclose(f1);  
fclose(f2);  
close(connfd);  
}  
return 0;  
}
```

//Client Program:

```
#include<stdio.h>  
#include<sys/socket.h>  
#include<netinet/in.h>  
#include<stdlib.h>  
#include<unistd.h>  
#include<string.h>  
#include<netdb.h>  
#include<arpa/inet.h>  
#include<sys/types.h>  
#include<time.h>  
#define MAX 100;  
main(int argc,char**argv)  
{  
int sockfd,n,connfd;  
char str[100],str1[100],s[2];  
struct sockaddr_in serv;  
if(argc!=5)
```

```
{
exit(0);
}
if(sockfd=socket(AF_INET,SOCK_STREAM,0)<0)
{
printf("\n Error ! Socket not created...\n");
exit (0);
}
bzero(&serv,sizeof(serv));
serv.sin_family=AF_INET;
serv.sin_port=htons(atoi(argv[2]));
if(inet_pton(AF_INET,argv[1],&serv.sin_addr)<=0)
{
printf("\n error in conversion of IP address from string to num\n"),
exit(0);
}
if(connect(sockfd,(struct sockaddr*)&serv,sizeof(serv))<0)
{
printf("\n Error!Conx not established...\n");
exit(0);
}
printf("\n connected...sending file name%s\n",argv[3]);
write(sockfd,argv[],strlen(argv[3]));
read(sockfd,s,3);
write(sockfd,argv[4],strlen(argv[4]));
str1[0]='\0';
while((n=read(sockfd,str,100))>0)
{
```

```
str[n]='\0';  
printf("%s\n",str);  
write(sockfd,str,strlen(str));  
}  
if(n<0)  
printf("\n Read error...\n");  
exit (0);  
}
```

EXPERIMENT NO: 10

NAME OF THE EXPERIMENT: UDP.

OBJECTIVE: Write a c program for UDP protocol.

RESOURCE: Turbo C

PROGRAM LOGIC: UDP provides a minimal, unreliable, best-effort, message-passing transport to applications and upper-layer protocols. Compared to other transport protocols, UDP and its UDP-Lite variant are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection establishment and teardown overheads and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to some applications, but has no inherent congestion control or reliability.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

Source Program:

```
/ Server side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT    8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
```

```
servaddr.sin_family = AF_INET; // IPv4
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);

// Bind the socket with the server address
if ( bind(sockfd, (const struct sockaddr *)&servaddr,
        sizeof(servaddr)) < 0 )
{
    perror("bind failed");
    exit(EXIT_FAILURE);
}

int len, n;
n = recvfrom(sockfd, (char *)buffer, MAXLINE,
        MSG_WAITALL, ( struct sockaddr *) &cliaddr,
        &len);
buffer[n] = '\0';
printf("Client : %s\n", buffer);
sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
        len);
printf("Hello message sent.\n");

return 0;
}
```

Output :

1. \$./server
2. Client : Hello from client
3. Hello message sent.
4. \$./client
5. Hello message sent.
6. Server : Hello from server

// Client side implementation of UDP client-server model

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 8080
#define MAXLINE 1024
```



```
// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &servaddr,
        sizeof(servaddr));
    printf("Hello message sent.\n");

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
        MSG_WAITALL, (struct sockaddr *) &servaddr,
        &len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);

    close(sockfd);
    return 0;
}
```

Output:

```
7. $ ./server
8. Client : Hello from client
9. Hello message sent.
10. $ ./client
11. Hello message sent.
12. Server : Hello from server
```