//C++ PROGRAM TO IMPLEMENT CRC (CYCLIC REDUNDANCY CHECK)//

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
        int i,j,n,g,a,ms[20],gen[20],b[20],q[20],s;
        clrscr();
        cout<<"\n\tTRANSMITTER SIDE\n";
        cout<<"\nEnter no. of data bits\n";
        cin>>n;
        cout<<"\nEnter data\n";
        for(i=0;i<n;i++)
        cin>>ms[i];
        cout<<"\nEnter size of generator\n";
        cin>>g;
        cout<<"\nEnter generator\n";
        for(j=0;j<g;j++)
        cin>>gen[j];
        cout<<"\n\nThe generated matrix is:\n\n";
        for(j=0;j<g;j++)
        cout<<gen[j];
        a=n+(g-1);
```

```
cout<<"\n\nThe appended matrix is:";
for(i=0;i<j;i++)
ms[n+i]=0;
for(i=0;i<a;i++)
cout<<ms[i];
for(i=0;i<n;i++)
q[i]=ms[i];
for(i=0;i<n;i++)
{
        if(ms[i]==0)
        {
                for(j=i;j<g+i;j++)
                {
                        ms[j]=ms[j]^0;
                }
        }
        else
        {
                ms[i]=ms[i]^gen[0];
                ms[i+1]=ms[i+1]^gen[1];
                ms[i+2]=ms[i+2]^gen[2];
                ms[i+3]=ms[i+3]^gen[3];
        }
```

```
        }
        cout<<"\n\nThe crc is:";
        for(i=n;i<a;i++)
        cout<<ms[i];
        s=n+a;
        for(i=n;i<s;i++)
        q[i]=ms[i];
        cout<<"\n\n";
        for(i=0;i<a;i++)
        cout<<q[i];
        getch();
}
```

//OUTPUT FOR CYCLIC REDUNDANCY CHECK//

TRANSMITTER SIDE

Enter no. of data bits

3

Enter data

5

6

4

Enter size of generator

5

Enter generator

8

9

1

4

6

The generated matrix is:

89146

The appended matrix is: 5640000

The crc is: 12540

56412540

//C++ PROGRAM TO IMPLEMENT VRC (VERTICAL REDUNDANCY CHECK)//

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
int binary(int);
void parity(int[]);
int arr[9],arr1[9];
int temp,temp1,i;
char chr;
void main()
{
        char chr1;
        clrscr();
        cout<<"Enter Data:";
        cin>>chr>>chr1;
        temp=chr;
        binary(temp);
        cout<<"\n ASCII value is:\n"<<temp;
        cout<<"\n Binary form:";
        for(i=0;i<8;i++)
        {
                arr1[i]=arr[i];
                cout<<arr[i];
```

```
        }
        cout<<"\n";
        parity(arr);
        temp1=chr1;
                binary(temp1);
        cout<<"\n ASCII value is:\n"<<temp1;
        cout<<"\n Binary form:";
        for(i=0;i<8;i++)
        {
                cout<<arr[i];
        }
        parity(arr);
        getch();
}
void parity(int a[])
{
        int count;
        count=0;
        for(i=0;i<8;i++)
        {
                if(a[i]==1)
                        count++;
        }
```

```
        if(count%2==0)
                a[8]=0;
        else
                a[8]=1;
        count=0;
        cout<<"\n Receiver side\n";
        cout<<"\n\n VRC:\n";
        for(i=0;i<9;i++)
        {
                if(i==8)
                        cout<<"|";
                        cout<<a[i];
        }
}
int binary(int x)
{
        int rem;
        int ctr=0,i=1;
        do
        {
                rem=x%2;
                arr[i]=rem;
                if(rem==1)
```

```
            {
                ctr++;
            }
            x=x/2;
            i++;
        }
        while(x!=0);
        if(ctr%2==0)
        {
            arr[0]=0;
        }
        else
        {
            arr[0]=1;
        }
        return(0);
}
```

//OUTPUT FOR VERTICAL REDUNDANCY CHECK//

Enter Data: 1 2 0 1

ASCII value is:

49

Binary form: 11000110

Receiver side

VRC:

11000110|0

ASCII value is:

50

Binary form:10100110

Receiver side

VRC:

10100110|0

//C++ PROGRAM TO IMPLEMENT BIT STUFFING//

```
#include<iostream.h>
#include<conio.h>
class Stuff
{
        private:
                int i,j,p,q,flag,data1[50],data2[50],cnt;
        public:
                void getinput();
                void bitstuff();
};
void Stuff::getinput()
{
        i=0;
        cout<<"\nEnter the original data (i.e. 0's & 1's):Type -1 At End"<<endl;
        while(data1[i]!=-1)
        {
                i++;
                cin>>data1[i];
        }
        cnt=i-1;
        cout<<"\nNumber of bits:"<<cnt<<"\n";
        cout<<"\nStuffed Data\n";
```

```
        for(j=1;j<=cnt;j++)
        cout<<data1[j]<<" ";
}
void Stuff::bitstuff()
{
        p=0;
        q=0;
        cout<<"\n\nData after stuffing\n";
        flag=0;
        while(q<=cnt)
        {
                q++;
                p++;
                data2[p]=data1[q];
                if(data1[q]==1)
                flag++;
                if(flag==5)
                {
                        p++;
                        data2[p]=0;
                        flag=0;
                }
        }
```

```
        cnt=p-1;
        for(i=1;i<=cnt;i++)
        cout<<data2[i]<<" ";
}
void main()
{
        clrscr();
        Stuff s;
        s.getinput();
        s.bitstuff();
        getch();
}
```

//OUTPUT FOR BIT STUFFING//

Enter the original data (i.e. 0's & 1's):Type -1 At End

0 1 1 0 1 0 1 0 1 1 1 0 0 0 -1

Number of bits:14

Stuffed Data

0 1 1 0 1 0 1 0 1 1 1 0 0 0

Data after stuffing

0 1 1 0 1 0 1 0 1 1 1 0 0 0

//C++ PROGRAM TO IMPLEMENT CHARACTER STUFFING//

```
#include<iostream.h>
#include<conio.h>
class C_Stuff
{
	private:
		int i,cnt,j;
		char str1[50],str2[70],str3[50];
	public:
		void send_data();
		void stuffing();
		void destuffing();
};
void C_Stuff::send_data()
{
	cout<<"\nEnter the data:Type -1 At End"<<endl;
	i=1;
	cin>>str1[i];
	while(str1[i]!='1')
	{
		i++;
		cin>>str1[i];
	}
```

```
        cnt=i-1;
        cout<<"\nData Sent By Network Layer"<<endl;
        for(i=1;i<=cnt;i++)
                cout<<str1[i]<<"  ";
                stuffing();
}
void C_Stuff::stuffing()
{
        j=1;
        cout<<"\n\nData After Character Stuffing"<<endl;
        for(i=1;i<=cnt;i++)
        {
                if(str1[i]=='e'||str1[i]=='E')
                {
                        str2[j]=str2[j+1]='E';
                        j=j+2;
                }
                else
                {
                        str2[j]=str1[i];
                        j++;
                }
        }
```

```
        cnt=j;
        str2[0]='S';
        str2[cnt]='E';
        for(i=0;i<=cnt;i++)
                cout<<str2[i]<<"  ";
        destuffing();
}
void C_Stuff::destuffing()
{
        i=1;j=1;
        cout<<"\n\nData After Character Destuffing"<<endl;
        while(i<=cnt)
        {
                if((str2[i]=='e'||str2[i]=='E')&&(str2[i+1]=='e'||str2[i+1]=='E'))
                {
                        str3[j]=str2[i];
                        j++;
                        i+=2;
                }
                else
                {
                        str3[j]=str2[i];
                        j++;
```

```
                    i++;
                }
        }
        cnt=j-1;
        for(i=1;i<cnt;i++)
        cout<<str3[i]<<"  ";
}
void main()
{
        clrscr();
        C_Stuff C;
        C.send_data();
        getch();
}
```

//OUTPUT FOR CHARACTER STUFFING//

Enter the data: Type -1 At End

Computer Networks -1

Data Sent By Network Layer

C o m p u t e r N e t w o r k s -

Data After Character Stuffing

S C o m p u t E E r N E E t w o r k s - E

Data After Character Destuffing

C o m p u t E r N E t w o r k s -

//C++ PROGRAM TO IMPLEMENT STOP AND WAIT ARQ PROTOCOL//

```
#include<iostream.h>
#include<conio.h>
class Protocol
{
        private:
                int i,j,f;
                char ch;
        public:
                void getdata();
                void stopwait();
};
void Protocol::getdata()
{
        cout<<"\nEnter the total number of frames you want to send:";
        cin>>f;
        if(f<=0)
                cout<<"\nFrames have been requested\n";
        else
                stopwait();
}
```

```
void Protocol::stopwait()
{
        i=0;
        j=0;
        while(i<f)
        {
                cout<<"\nFrame"<<i<<"is sent\n";
                cout<<"\nIs acknowledgement"<<j<<"received?(Y/N):";
                cin>>ch;
                if(ch=='y'||ch=='Y')
                {
                        i++;
                        j++;
                }
                else
                        cout<<"\nSend again...\n";
        }
        cout<<"\nInformation sent successfully!!\n";
}
```

```
void main()
{
        clrscr();
        Protocol p;
        p.getdata();
        getch();
}
```

//OUTPUT FOR STOP AND WAIT PROTOCOL//

Enter the total number of frames you want to send: 3

Frame 0 is sent

Is acknowledgement 0 received?(Y/N):y

Frame 1 is sent

Is acknowledgement 1 received?(Y/N):y

Frame 2 is sent

Is acknowledgement 0 received?(Y/N):n

Send again…

Frame 2 is sent

Is acknowledgement 2 received?(Y/N):y

Information sent successfully!!

//C++ PROGRAM TO IMPLEMENT GO-BACK-N ARQ PROTOCOL//

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<stdlib.h>
int i,n,j,f;
char c;
void cal();
class arq
{
        public:
                void get();
};
void arq::get()
{
        randomize();
        for(i=1;i<=n;i++)
        {
                cout<<"\nFrame"<<i<<"is sending\n";
                cal();
        }
```

```
for(i=1;i<=n;i++)
{
        cout<<"\nAck"<<i<<"is received\n";
        cal();
}
cout<<"\nDo you want to send any other frame(y/n):";
cin>>c;
if(c=='y'||c=='Y')
{
        cout<<"Enter the number of frames you want to send:";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cout<<"\nFrame"<<i+1<<"is sending\n";
                cal();
        }
        f=random(n);
        for(i=0;i<n;i++)
        {
                if(i==f)
                {
                        cout<<"\nFrame"<<f+1<<"is lost\n";
                        cal();
```

```
                cout<<"\nPlease resend it\n";
                cal();
                cout<<"\nAck"<<i+1<<"is not received\n";
            }
            else
            {
                cout<<"\nAck"<<i+1<<"is received\n";
                cal();
            }
        }
}
else
{
        cout<<"\nInformation sent successfully!!\n";
        cal();
        exit(0);
}
cout<<"Please resend all frames again\n";
cal();
```

```
        for(i=0;i<n;i++)
        {
                cout<<"\nFrame"<<i+1<<"is resending\n";
                cal();
        }
        cout<<"\n\n";
        for(i=1;i<=n;i++)
        {
                cout<<"\nAck"<<i<<"is received\n";
                cal();
        }
        cout<<"\nInformation sent successfully!!\n";
}
void cal()
{
        for(j=0;j<3;j++)
        {
                sleep(j);
                cout<<"       ";
        }
}
```

```
void main()
{
        clrscr();
        cout<<"Enter the number of frames:";
        cin>>n;
        arq ob;
        ob.get();
        getch();
}
```

//OUTPUT FOR GO-BACK-N//

Enter the number of frames: 3

Frame 1 is sending

Frame 2 is sending

Frame 3 is sending

Ack 1 is received

Ack 2 is received

Ack 3 is received

Do you want to send any other frame(y/n):y

Enter the number of frames you want to send: 2

Frame 1 is sending

Frame 2 is sending

Ack 1 is received

Frame 2 is lost

Please resend it

Ack 2 is not received

Please resend all frames again

Frame 1 is sending

Frame 2 is sending

Ack 1 is received

Ack 2 is received

Information sent successfully!!

//C++ PROGRAM TO IMPLEMENT SELECTIVE-REPEAT ARQ PROTOCOL//

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<stdlib.h>
int i,n,j,f;
char c;
void cal();
class arq
{
        public:
                void get();
};
void arq::get()
{
        randomize();
        for(i=1;i<=n;i++)
        {
                cout<<"\nFrame"<<i<<"is sending\n";
                cal();
        }
```

```
for(i=1;i<=n;i++)
{
        cout<<"\nAck"<<i<<"is received\n";
        cal();
}
cout<<"\nDo you want to send any other frame(y/n):";
cin>>c;
if(c=='y'||c=='Y')
{
        cout<<"Enter the number of frames you want to send:";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cout<<"\nFrame"<<i+1<<"is sending\n";
                cal();
        }
        f=random(n);
        for(i=0;i<n;i++)
        {
                if(i==f)
                {
                        cout<<"\nFrame"<<f+1<<"is lost\n";
                        cal();
```

```
                cout<<"\nPlease resend it\n";
                cal();
                cout<<"\nAck"<<i+1<<"is not received\n";
            }
            else
            {
                cout<<"\nAck"<<i+1<<"is received\n";
                cal();
            }
        }
}
else
{
        cout<<"\nInformation sent successfully!!\n";
        cal();
        exit(0);
}
cout<<"Please resend all frames again\n";
cal();
for(i=0;i<n;i++)
{
        cout<<"\nFrame"<<i+1<<"is resending\n";
        cal();
```

```
        }
        cout<<"\n\n";
        for(i=1;i<=n;i++)
        {
                cout<<"\nAck"<<i<<"is received\n";
                cal();
        }
        cout<<"\nInformation sent successfully!!\n";
}
void cal()
{
        for(j=0;j<3;j++)
        {
                sleep(j);
                cout<<"       ";
        }
}
void main()
{
        clrscr();
        cout<<"Enter the number of frames:";
```

```
        cin>>n;
        arq ob;
        ob.get();
        getch();
}
```

//OUTPUT FOR SELECTIVE-REPEAT ARQ PROTOCOL//

Enter the number of frames: 2

Frame 1 is sending

Frame 2 is sending

Ack 1 is received

Ack 2 is received

Do you want to send any other frame(y/n):y

Enter the number of frames you want to send: 3

Frame 1 is sending

Frame 2 is sending

Frame 3 is sending

Ack 1 is received

Frame 2 is lost

Please resend it

Ack 2 is not received

Ack 3 is received

Please resend lost frame again

Frame 2 is sending

Ack 2 is received

Information sent successfully!!

//C++ PROGRAM TO IMPLEMENT DIJKSTRA's ALGORITHM//

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<math.h>
#define IN 99
#define N 6
int dijkstra(int cost[][N],int source,int target);
int dijkstra(int cost[][N],int source,int target)
{
        int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j;
        char path[N];
        for(i=1;i<N;i++)
        {
                dist[i]=IN;
                prev[i]=-1;
        }
        start=source;
        selected[start]=1;
        dist[start]=0;
        while(selected[target]==0)
        {
```

```
        min=IN;
        m=0;
        for(i=1;i<N;i++)
        {
                d=dist[start]+cost[start][i];
                if(d<dist[i]&&selected[i]==0)
                {
                        dist[i]=d;
                        prev[i]=start;
                }
                if(min>dist[i]&&selected[i]==0)
                {
                        min=dist[i];
                        m=i;
                }
        }
        start=m;
        selected[start]=1;
}
start=target;
j=0;
```

```
        while(start!=-1)
        {
                path[j++]=start+65;
                start=prev[start];
        }
        path[j]='\0';
        strrev(path);
        cout<<path;
        return dist[target];
}
void main()
{
        int cost[N][N],i,j,w,ch,co;
        int source,target,x,y;
        clrscr();
        cout<<"\tShortest Path Algorithm (DIJKSTRA's ALGORITHM)\n\n";
        for(i=1;i<N;i++)
        for(j=1;j<N;j++)
        cost[i][j]=IN;
        for(x=1;x<N;x++)
        {
```

```
            for(y=x+1;y<N;y++)
            {
                cout<<"Enter the weight of the path between node"<<x<<"and"
                <<y<<endl;
                cin>>w;
                cost[x][y]=cost[y][x]=w;
            }
            cout<<"\n";
        }
        cout<<"\nEnter the source:";
        cin>>source;
        cout<<"\nEnter the target";
        cin>>target;
        co=dijkstra(cost,source,target);
        cout<<"\nShortest Path:"<<co;
        getch();
}
```

//OUTPUT FOR DIJKSTRA's ALGORITHM//

Shortest Path Algorithm (DIJKSTRA's ALGORITHM)

Enter the weight of the path between node 1 and 2

2

Enter the weight of the path between node 1 and 3

3

Enter the weight of the path between node 1 and 4

5

Enter the weight of the path between node 1 and 5

6

Enter the weight of the path between node 2 and 3

9

Enter the weight of the path between node 2 and 4

7

Enter the weight of the path between node 2 and 5

1

Enter the weight of the path between node 3 and 4

2

Enter the weight of the path between node 3 and 5

0

Enter the weight of the path between node 4 and 5

5

Enter the source:

1

Enter the target:

5

Shortest path: 3

//C++ PROGRAM TO FIND THE GIVEN DOTTED DECIMAL IP ADDRESS IS VALID OR NOT//

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<math.h>
#include<stdio.h>
int isValidIpAddress(char *st)
{
        int num,i,len;
        char *ch;
        int quadsCnt=0;
        cout<<"Split IP:"<<st<<"\n";
        len=strlen(st);
        if(len<7||len>15)
                return 0;
        ch=strtok(st,".");
        while(ch!=NULL)
        {
                quadsCnt++;
                cout<<"Id"<<quadsCnt<<"is"<<ch<<"\n";
                num=0;
                i=0;
```

```
        while(ch[i]!='\0')
        {
                num=num*10;
                num=num+(ch[i]-'0');
                i++;
        }
        if(num<0||num>255)
        {
                cout<<"Not a valid IP address"<<"\n";
                return 0;
        }
        if((quadsCnt==1&&num==0)||(quadsCnt==4&&num==0))
        {
                cout<<"Not a valid IP address"<<"\n";
                return 0;
        }
        ch=strtok(NULL,".");
}
if(quadsCnt!=4)
{
        return 0;
}
```

```
        return 1;
}
void main()
{
        char *st;
        clrscr();
        cout<<"Enter the IP address:"<<"\n";
        cin>>st;
        if(isValidIpAddress(st))
        {
                cout<<"The given IP is a valid IP address"<<"\n";
        }
        else
        {
                cout<<"The given IP is not a valid IP address"<<"\n";
        }
        getch();
}
```

//OUTPUT FOR IP ADDRESS IS VALID OR NOT//

1) Enter the IP address:
   193.45.34.67
   Split IP: 193.45.34.67
   Id 1 is 193
   Id 2 is 45
   Id 3 is 34
   Id 4 is 67
   The given IP is a valid IP address

2) Enter the IP address:
   635.186.16.62
   Split IP: 635.186.16.62
   Id 1 is 635
   Not a valid IP address
   The given IP is not a valid IP address

//C++ PROGRAM TO FIND A CLASS IN A GIVEN DOTTED DECIMAL IP ADDRESS//

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<math.h>
#include<stdio.h>
int ValidIpAddress(char *st)
{
        int num,i,len;
        char *ch;
        int quadsCnt=0;
        len=strlen(st);
        if(len<7||len>15)
                return 0;
        ch=strtok(st,".");
        while(ch!=NULL)
        {
                quadsCnt++;
                num=0,i=0;
                while(ch[i]!='\0')
                {
                        num=num*10;
                        num=num+(ch[i]-'0');
```

```
        i++;
}
if(num<0||num>255)
{
        cout<<"Not a valid IP address"<<"\n";
        return 0;
}
if((quadsCnt==1&&num==0)||(quadsCnt==4&&num==0))
{
        cout<<"Not a valid IP address"<<"\n";
        return 0;
}
if(quadsCnt==1)
{
        if(num>=1&&num<=127)
        {
                cout<<"Class A\n";
        }
        else if(num>=128&&num<=191)
        {
                cout<<"Class B\n";
        }
        else if(num>=192&&num<=223)
```

```
                {
                        cout<<"Class C\n";
                }
                else if(num>=224&&num<=239)
                {
                        cout<<"Class D\n";
                }
                else if(num>=240&&num<=255)
                {
                        cout<<"Class E\n";
                }
        }
        ch=strtok(NULL,".");
    }
    if(quadsCnt!=4)
    {
        return 0;
    }
    return 1;
}
void main()
{
    char *st;
```

```
        clrscr();
        cout<<"Enter the IP address:"<<"\n";
        cin>>st;
        if(ValidIpAddress(st))
        {
                cout<<"The given IP is a valid IP address"<<"\n";
        }
        else
        {
                cout<<"The given IP is not a valid IP address"<<"\n";
        }
        getch();
}
```

//OUTPUT FOR IP ADDRESS CLASS//

1) Enter the IP address:
   192.1.15.56
   Class C
   The given IP is a valid IP address

2) Enter the IP address:
   300.1.15.506
   Not a valid IP address
   The given IP is not a valid IP address

//C++ PROGRAM TO FIND THE SHORTEST PATH//

```
#include<iostream.h>
#include<conio.h>
#define MAX_NODES 20
#define INFINITY 10000
struct state
{
        int pre,len;
        char label;
}state[MAX_NODES];
class Shortpath
{
        private:
                int i,j,n,f,s,d,k;
                int min,path[20],dist[MAX_NODES][MAX_NODES];
                struct state *p;
        public:
                void getdata();
                void find_shortpath(int,int,int[]);
```

```
};
void Shortpath::getdata()
{
        cout<<"\nEnter the number of nodes:";
        cin>>n;
        cout<<"\nNodes:";
        cout<<"\nEnter the distance between various nodes:";
        cout<<"\n";
        for(i=1;i<=n;i++)
        cout<<"\t"<<i;
        cout<<"\n";
        for(i=1;i<=n;i++)
        {
                cout<<i<<"\t";
                for(j=1;j<=n;j++)
                        cin>>dist[i][j];
        }
        cout<<"\nEnter the source node:\n";
        cin>>s;
```

```
        cout<<"\nEnter the destination node:\n";
        cin>>d;
        find_shortpath(s,d,path);
}
void Shortpath::find_shortpath(int s,int d,int path[])
{
        for(p=&state[1];p<=&state[n];p++)
        {
                p->pre=-1;
                p->len=INFINITY;
                p->label='t';
        }
        state[d].len=0;
        state[d].label='p';
        k=d;
        do
        {
                for(i=1;i<=n;i++)
                if(dist[k][i]!=0&&state[i].label=='t')
```

```
        {
                state[i].pre=k;
                state[i].len=state[k].len+dist[k][i];
        }
        k=0;
        min=INFINITY;
        for(i=1;i<=n;i++)
        if(state[i].label=='t'&&state[i].len<min)
        {
                min=state[i].len;
                k=i;
        }
        state[k].label='p';
}
while(k!=s);
i=0;
k=s;
cout<<"\nShortest Path is\n";
```

```
        do
        {
                path[i=i+1]=k;
                cout<<k<<"  ";
                k=state[k].pre;
        }
        while(k>=0);
}
void main()
{
        clrscr();
        Shortpath s;
        s.getdata();
        getch();
}
```

//OUTPUT FOR SHORTEST PATH//

Enter the number of nodes: 4

Nodes:

Enter the distance between various nodes:

	1	2	3	4
1	4	6	7	6
2	3	2	1	3
3	5	4	3	2
4	1	2	4	5

Enter the source node: 1

Enter the destination node: 3

Shortest Path is

1 4 3

//C++ PROGRAM TO IMPLEMENT PUBLIC-KEY ENCRYPTION//

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int i,key,pkey,len=0;
        char string[40],d[20],e[20];
        clrscr();
        cout<<"Enter the string:\n";
        gets(string);
        cout<<"Enter the public key to encrypt:\n";
        cin>>key;
        cout<<"\nEncryption\n";
        len=strlen(string);
        for(i=0;i<len;i++)
        {
                e[i]=string[i]+key;
                cout<<e[i];
        }
        cout<<"\n\n Enter only the private key to decrypt\n";
        cin>>pkey;
```

```
        if(pkey==2)
        {
                cout<<"\nDecryption\n";
                for(i=0;i<len;i++)
                {
                        d[i]=e[i]-key;
                        cout<<d[i];
                }
        }
        else
        cout<<"Authentication fails\n";
        getch();
}
```

//OUTPUT FOR PUBLIC-KEY ENCRYPTION//

1. Enter the string:
   network
   Enter the public key to encrypt:
   2
   Encryption
   pgvyqtm
   Enter only the private key to decrypt:
   2
   Decryption
   network
2. Enter the string:
   applications
   Enter the public key to encrypt:
   3
   Encryption
   dssolfdwlrq
   Enter only the private key to decrypt:
   4
   Authentication fails

//C++ PROGRAM TO IMPLEMENT COLUMNAR TRANSPOSITION//

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<stdio.h>
class col_trans
{
        int i,c1,c,c2,p1[30],pr;
        char ch1,str1[20],str2[50],string[30][30];
        public:
                void getdata();
                void process();
};
int main()
{
      col_trans c;
      clrscr();
      c.getdata();
      getch();
      exit(0);
      return 1;
}
```

```
void col_trans::getdata()
{
	cout<<"\n\nProgram to implement columnar transposition\n";
	cout<<"Enter the keyword:";
	c1=1;
	do
	{
		ch1=getchar();
		str1[c1]=ch1;
		c1++;
	}
	while(ch1!='\n');
	c1=c1-2;
	cout<<"\nNumber of characters="<<c1<<"\n";
	process();
}
void col_trans::process()
{
	int j=1;
	for(int ch=65;ch<=122;ch++)
	{
		if(ch<90||ch>=97)
		for(i=1;i<=c1;i++)
```

```
        {
                if(str1[i]==ch)
                {
                        p1[i]=j;
                        j++;
                        break;
                }
        }
}
cout<<"\n";
cout<<"Enter the plain text:";
c2=1;
do
{
        ch1=getchar();
        str2[c2]=ch1;
        c2++;
}
while(ch1!='\n');
c2=c2-2;
int r=c2/c1;
int p=1;
for(i=1;i<=r+1;i++)
```

```
        for(j=1;j<=c1;j++)
        {
                if(p<=c2)
                {
                        string[i][j]=str2[p];
                        p++;
                }
                else
                        string[i][j]=' ';
        }
        cout<<"\n\n";
        cout<<"Coded text\n\n";
        for(i=1;i<=c1;i++)
        cout<<str1[i]<<" ";
        cout<<"\n";
        for(i=1;i<=r+1;i++)
        {
                for(j=1;j<=c1;j++)
                cout<<string[i][j]<<" ";
                cout<<"\n";
        }
}
```

//OUTPUT FOR COLUMNAR TRANSPOSITION//

Program to implement columnar transposition

Enter the keyword: network

Number of characters = 7

Enter the plain text: computer

Coded text

n e t w o r k

c o m p u t e

r

//C++ PROGRAM TO IMPLEMENT ENCRYPTION ALGORITHM//

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class Encrypt
{
        private:
                int i,len;
                char s[10],e[10];
        public:
                void input();
                void encryption();
};
void Encrypt::input()
{
        cout<<"\n Enter the string to encrypt:\n";
        gets(s);
        while(s[i]!='\0')
        i=i+1;
```

```
        len=10;
        encryption();
}
void Encrypt::encryption()
{
        cout<<"\n After Encryption:\n";
        for(i=0;i<len;i++)
        {
                e[i]=s[i]+2;
                cout<<e[i];
        }
}
void main()
{
        clrscr();
        Encrypt E;
        E.input();
        getch();
}
```

//OUTPUT FOR ENCRYPTION//

Enter the string to encrypt:

Networking

After Encryption:

Pgvyqtmkpi

//C++ PROGRAM TO IMPLEMENT PARITY CHECK//

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
int main()
{
        int bin[7],x=0,y;
        clrscr();
        cout<<"Enter binary number 1 or 0\n\n";
        for(int z=0;z<7;z++)
        {
                cout<<"Binary"<<"("<<z+1<<"):";
                cin>>bin[z];
                if(bin[z]==1)
                        x++;
        }
        cout<<"\n\nResult:\n";
        y=x%2;
        if(y==0)
        {
                cout<<"Even Parity=0\n";
                cout<<"Odd Parity=1\n";
        }
```

```
        else
        {
                cout<<"Even Parity=1\n";
                cout<<"Odd Parity=0\n";
        }
        getch();
        return 0;
}
```

//OUTPUT FOR PARITY CHECK//

Enter binary number 1 or 0

Binary(1):1

Binary(2):1

Binary(3):0

Binary(4):1

Binary(5):0

Binary(6):0

Binary(7):1

Result:

Even Parity=0

Odd Parity=1