**Assignment Number: 1**

**Asian School Of Management**

**Name: Pralhad Khanal**

**Class: B.Sc.CSIT V Sem**

**Roll Number: 29**

**Question 1) Implement Data Encryption Standard in Any Programming Language**

#include<bits/stdc++.h>

using namespace std;

string hex2bin(string p)//hexadecimal to binary

{

    string ap="";

    int l=p.length();

    for(int i=0;i<l;i++)

    {

        string st="";

        if(p[i]>='0'&&p[i]<='9')

        {

            int te=int(p[i])-48;

            while(te>0)

            {

                st+=char(te%2+48);

                te/=2;

            }

            while(st.length()!=4)

                st+='0';

            for(int j=3;j>=0;j--)

                ap+=st[j];

        }

        else

        {

            int te=p[i]-'A'+10;

            while(te>0)

            {

                st+=char(te%2+48);

                te/=2;

            }

            for(int j=3;j>=0;j--)

                ap+=st[j];

        }

    }

    return ap;

}

int main()

{

        string p,l,r,ap="",ke,kp,rtem;

        pre:;

        //Input of the plain text and the key

        cout<<"Enter the plain text in hexadecimal form (16bit) \n";

        cin>>p;

        if(p.length()!=16)

        {

            cout<<"enter all the bits\n";

            goto pre;

        }

        for(int i=0;i<16;i++)

        {

            if((p[i]>='0'&&p[i]<='9')||(p[i]>='A'&&p[i]<='F'))

                ;

            else

            {

                cout<<"Not a valid hexadecimal string\n";

                goto pre;

            }

        }

        int key1[64];

        pr:;

                cout<<"Enter the key in hexadecimal form (16bit) \n";

                cin>>ke;

         if(ke.length()!=16)

         {

            cout<<"enter all the key bits\n";

            goto pr;

         }

        for(int i=0;i<16;i++)

        {

            if((ke[i]>='0'&&ke[i]<='9')||(ke[i]>='A'&&ke[i]<='F'))

                ;

            else

            {

                cout<<"Not a valid hexadecimal key string\n";

                goto pr;

            }

        }

        //Input is completed

        p=hex2bin(p);

        kp=hex2bin(ke);

        for(int i=0;i<64;i++)

            key1[i]=kp[i]-'0';

    //getting 56 bit key from 64 bit using the parity bits

    int keyp[56]={57  , 49   , 41   ,33 ,   25   , 17 ,   9,

               1 ,  58    ,50  , 42 ,   34  ,  26  , 18,

              10 ,   2   , 59 ,  51  nbsp;,  43  ,  35  , 27,

              19  , 11   ,  3  , 60  ,  52  ,  44  , 36,

              63  , 55  ,  47  , 39  ,  31   , 23  , 15,

               7  , 62   , 54  , 46  ,  38  ,  30  , 22,

              14  ,  6   , 61  , 53  ,  45   , 37  , 29,

              21  , 13  ,   5  , 28   , 20  ,  12  ,  4};

    //D box configuration each 28 to 24 bit

    int key2[48]={14,    17,   11,    24,     1,    5,

    3   , 28 ,  15    , 6  ,  21  , 10 ,

    23    ,19  , 12   ,  4 ,  26  ,  8,

    16     ,7  , 27  ,  20  ,  13  ,  2,

    41   , 52 ,  31  ,  37  ,  47  , 55,

    30   , 40  , 51  ,  45  ,  33 , 48,

    44    ,49 ,  39  ,  56  ,  34 ,  53,

    46    ,42 ,  50  ,  36 ,   29 ,  32};

    int key[16][48],keyl[28],keyr[28],nshift,temp1,temp2,pkey[56];

    l=p.substr(0,32);

    r=p.substr(32,32);

    //key generation

    int i,t=1,j,row,col,temp,round=16;

    j=0;

    for(i=0;i<56;i++)

        pkey[i]=key1[keyp[i]-1];

    for(i=0;i<28;i++)

        keyl[i]=pkey[i];

    for(i=0;i<28;i++)

        keyr[i]=pkey[i+28];

    //round key generation

    for(i=0;i<16;i++)

    {

        if(i==0||i==1||i==8||i==15)

            nshift=1;

        else

            nshift=2;

        while(nshift--)

        {

            temp1=keyl[0];

            temp2=keyr[0];

            for(j=0;j<27;j++)

            {

                keyl[j]=keyl[j+1];

                keyr[j]=keyr[j+1];

            }

            keyl[27]=temp1;

            keyr[27]=temp2;

        }

        for(j=0;j<24;j++)

            key[i][j]=keyl[key2[j]-1];

        for(j=24;j<48;j++)

            key[i][j]=keyr[key2[j]-1-28];

    }

    //straight pbox

    int per[32]={16 ,  7 , 20  ,21,

                29 , 12 , 28 , 17,

                1 , 15,  23,  26,

                5 , 18 , 31  ,10,

                2  , 8 , 24 , 14,

                32  ,27,   3 ,  9,

                19  ,13,  30,   6,

                22 , 11  , 4 , 25};

    //sbox configuration 6 bit to 4 bit

    int s[8][4][16]=

    {{

        14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,

        0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,

        4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,

        15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13

    },

    {

        15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,

        3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,

        0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,

        13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9

    },

    {

        10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,

        13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,

        13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,

        1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12

    },

    {

        7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,

        13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,

        10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,

        3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14

    },

    {

        2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,

        14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,

        4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,

        11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3

    },

    {

        12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,

        10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,

        9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,

        4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13

    },

    {

        4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,

        13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,

        1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,

        6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12

    },

    {

        13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,

        1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,

        7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,

        2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11

    }};

    //DES Encryption

         while(round--)

         {

            rtem=r;

            t=1;

            string ep="",xorout="",sout="",soutt;

            //the expansion P box

            ep+=r[31];

            for(i=0;i<32;i++)

            {

                if((t+1)%6==0)

                {

                        ep+=r[4\*((t+1)/6)];

                        t++;

                }

                if(t%6==0&&i!=0)

                {

                    ep+=r[4\*(t/6)-1];

                    t++;

                }

                ep=ep+r[i];

                t++;

            }

            ep+=r[0];

            //Key xor with output of expansion p box

            for(i=0;i<48;i++)

                xorout+=char(((int(ep[i])-48)^key[16-round-1][i])+48);

            //sbox compression 48bit to 32 bit

            for(i=0;i<48;i+=6)

            {

                    row=(int(xorout[i+5])-48)+(int(xorout[i])-48)\*2;

                    col= (int(xorout[i+1])-48)\*8+(int(xorout[i+2])-48)\*4+(int(xorout[i+3])-48)\*2+(int(xorout[i+4])-48);

                    temp=s[i/6][row][col];

                    soutt="";

                    while(temp>0)

                    {

                        soutt+=char(temp%2+48);

                        temp/=2;

                    }

                    while(soutt.length()!=4)

                        soutt+='0';

                    for(j=soutt.length()-1;j>=0;j--)

                        sout+=soutt[j];

            }

            //straight pbox that is permutation of the sbox output

            char pc[32];

            for(i=0;i<32;i++)

                pc[i]=sout[per[i]-1];

            r="";

            for(i=0;i<32;i++)

                r+=char(((int(pc[i])-48)^(int(l[i])-48))+48);

            l=rtem;

            cout<<"Output after Round"<<16-round<<endl;

            string cip="";

            for(i=0;i<32;i+=4)

            {

                int te;

                te=(int(l[i])-48)\*8+(int(l[i+1])-48)\*4+(int(l[i+2])-48)\*2+(int(l[i+3])-48);

                if(te<10)

                    cip+=char(te+48);

                else

                    cip+=char(te+55);

            }

            for(i=0;i<32;i+=4)

            {

                int te;

                te=(int(r[i])-48)\*8+(int(r[i+1])-48)\*4+(int(r[i+2])-48)\*2+(int(r[i+3])-48);

                if(te<10)

                    cip+=char(te+48);

                else

                    cip+=char(te+55);

            }

            cout<<cip<<endl;

        }

//test case

/\*

PLAIN=

14A7D67818CA18AD

KEY=

AABB09182736CCDD

\*/

    //DES decryption

    round=16;

    string ltem;

    while(round--)

    {

        ltem=l;

        t=1;

        string ep="",xorout="",sout="",soutt;

        //the expansion P box

        ep+=l[31];

        for(i=0;i<32;i++)

        {

            if((t+1)%6==0)

            {

                ep+=l[4\*((t+1)/6)];

                t++;

            }

            if(t%6==0&&i!=0)

            {

                ep+=l[4\*(t/6)-1];

                t++;

            }

            ep=ep+l[i];

            t++;

        }

        ep+=l[0];

        //Key xor with output of expansion p box

        for(i=0;i<48;i++)

            xorout+=char(((int(ep[i])-48)^key[round][i])+48);

        //sbox compression 48bit to 32 bit

        for(i=0;i<48;i+=6)

        {

            row=(int(xorout[i+5])-48)+(int(xorout[i])-48)\*2;

            col= (int(xorout[i+1])-48)\*8+(int(xorout[i+2])-48)\*4+(int(xorout[i+3])-48)\*2+(int(xorout[i+4])-48);

            temp=s[i/6][row][col];

            soutt="";

            while(temp>0)

            {

                soutt+=char(temp%2+48);

                temp/=2;

            }

            while(soutt.length()!=4)

                soutt+='0';

            for(j=soutt.length()-1;j>=0;j--)

                sout+=soutt[j];

        }

        //straight pbox that is permutation of the sbox output

        char pc[32];

        for(i=0;i<32;i++)

            pc[i]=sout[per[i]-1];

        l="";

        for(i=0;i<32;i++)

            l+=char(((int(pc[i])-48)^(int(r[i])-48))+48);

        r=ltem;

        cout<<"Decrypted Output after Round"<<16-round<<endl;

        string cip="";

        for(i=0;i<32;i+=4)

        {

            int te;

            te=(int(l[i])-48)\*8+(int(l[i+1])-48)\*4+(int(l[i+2])-48)\*2+(int(l[i+3])-48);

            if(te<10)

                cip+=char(te+48);

            else

                cip+=char(te+55);

        }

        for(i=0;i<32;i+=4)

        {

            int te;

            te=(int(r[i])-48)\*8+(int(r[i+1])-48)\*4+(int(r[i+2])-48)\*2+(int(r[i+3])-48);

            if(te<10)

                cip+=char(te+48);

            else

                cip+=char(te+55);

        }

        cout<<cip<<endl;

    }

        system("pause");

}