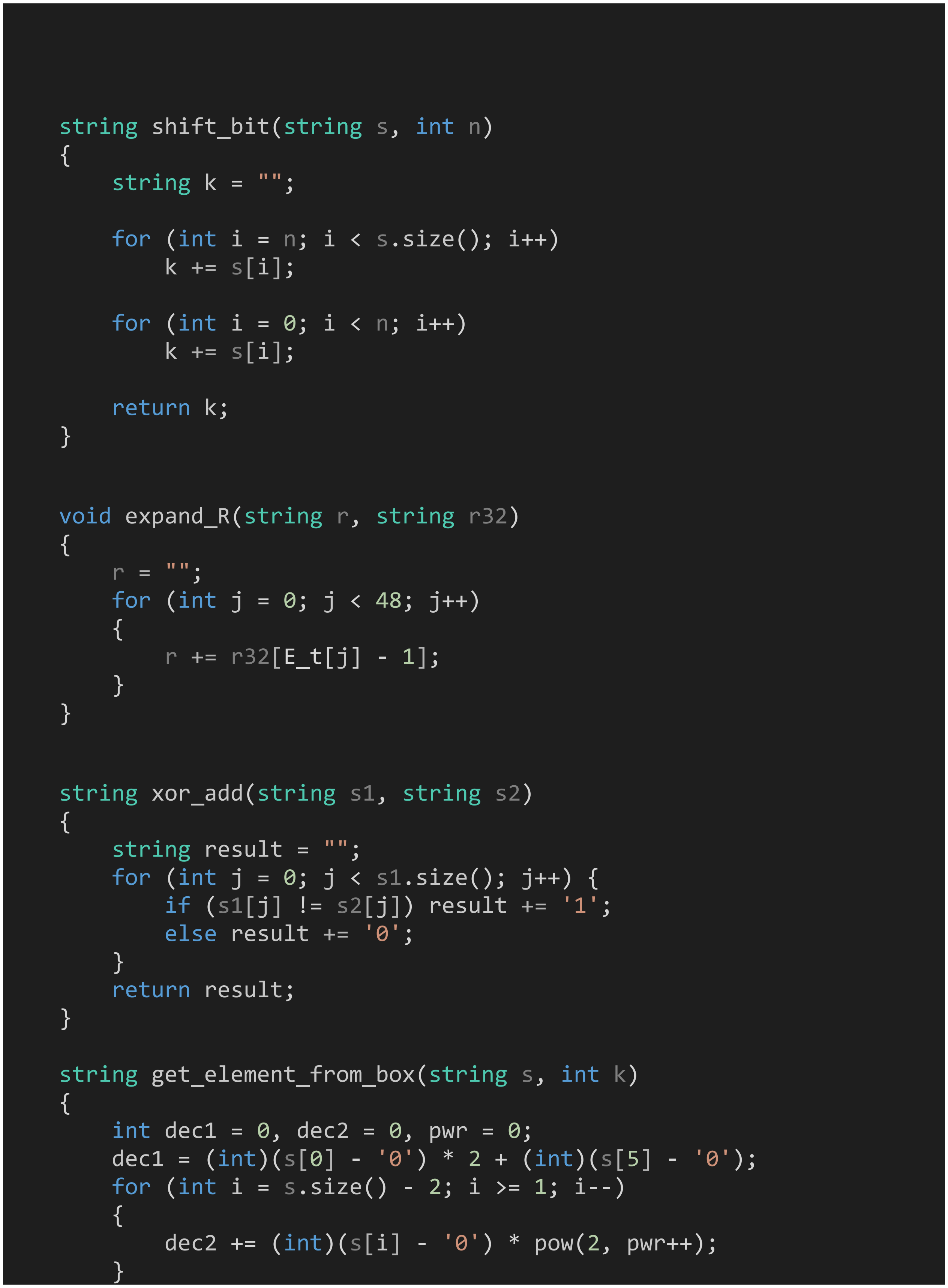
```cpp
#include <iostream>
#include <string>
#include <cmath>

using namespace std;


string Bin_to_Hex(string s);
string Hex_to_Bin(string s);
string Dec_to_Bin(int n);

class DES_Encryption
{



    const int pc_1[56] = {  57 ,49 ,41 ,33 ,25 ,17 ,9  ,
                1  ,58 ,50 ,42 ,34 ,26 ,18 ,
                10 ,2  ,59 ,51 ,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 };

    int num_leftShift[16] = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
2, 2, 1 };

    const int pc_2[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 };



const int IP_t[64] = {  58 ,50 ,42 ,34 ,26 ,18 ,10 ,2 ,
                60 ,52 ,44 ,36 ,28 ,20 ,12 ,4 ,
                62 ,54 ,46 ,38 ,30 ,22 ,14 ,6 ,
                64 ,56 ,48 ,40 ,32 ,24 ,16 ,8 ,
                57 ,49 ,41 ,33 ,25 ,17 ,9  ,1 ,
                59 ,51 ,43 ,35 ,27 ,19 ,11 ,3 ,
                61 ,53 ,45 ,37 ,29 ,21 ,13 ,5 ,
                63 ,55 ,47 ,39 ,31 ,23 ,15 ,7 };

const int E_t[48] = {   32 ,1  ,2  ,3  ,4  ,5  ,
                4  ,5  ,6  ,7  ,8  ,9  ,
                8  ,9  ,10 ,11 ,12 ,13 ,
                12 ,13 ,14 ,15 ,16 ,17 ,
                16 ,17 ,18 ,19 ,20 ,21 ,
                20 ,21 ,22 ,23 ,24 ,25 ,
                24 ,25 ,26 ,27 ,28 ,29 ,
                28 ,29 ,30 ,31 ,32 ,1 };

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 }
    }
};

const int P[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 };

const int P_1[64] = {   40 ,8  ,48 ,16 ,56 ,24 ,64 ,32 ,
            39 ,7  ,47 ,15 ,55 ,23 ,63 ,31 ,
            38 ,6  ,46 ,14 ,54 ,22 ,62 ,30 ,
            37 ,5  ,45 ,13 ,53 ,21 ,61 ,29 ,
            36 ,4  ,44 ,12 ,52 ,20 ,60 ,28 ,
            35 ,3  ,43 ,11 ,51 ,19 ,59 ,27 ,
            34 ,2  ,42 ,10 ,50 ,18 ,58 ,26 ,
            33 ,1  ,41 ,9  ,49 ,17 ,57 ,25 };
```

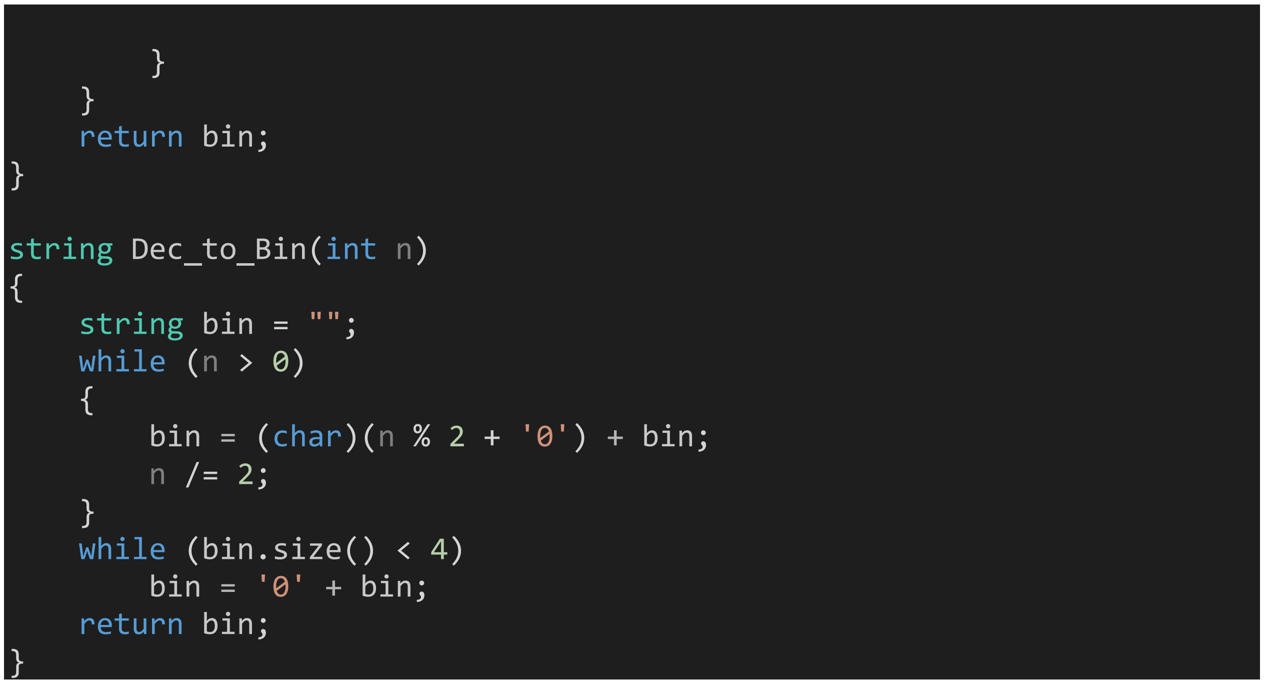


```cpp
string shift_bit(string s, int n)
{
    string k = "";

    for (int i = n; i < s.size(); i++)
        k += s[i];

    for (int i = 0; i < n; i++)
        k += s[i];

    return k;
}


void expand_R(string r, string r32)
{
    r = "";
    for (int j = 0; j < 48; j++)
    {
        r += r32[E_t[j] - 1];
    }
}


string xor_add(string s1, string s2)
{
    string result = "";
    for (int j = 0; j < s1.size(); j++) {
        if (s1[j] != s2[j]) result += '1';
        else result += '0';
    }
    return result;
}

string get_element_from_box(string s, int k)
{
    int dec1 = 0, dec2 = 0, pwr = 0;
    dec1 = (int)(s[0] - '0') * 2 + (int)(s[5] - '0');
    for (int i = s.size() - 2; i >= 1; i--)
    {
        dec2 += (int)(s[i] - '0') * pow(2, pwr++);
    }
```

```
        return Dec_to_Bin(S[k][dec1][dec2]);
    }


public:

    void encrypt(const string& plain_txt, const string& key)
    {


        string key_64 = Hex_to_Bin(key);



        string key_56 = "";
        string key_firstHalf = "", key_secondHalf = "";

        for (int i = 0; i < 56; i++)
            key_56 += key_64[pc_1[i] - 1];

        for (int i = 0; i < 28; i++)
            key_firstHalf += key_56[i];

        for (int i = 28; i < 56; i++) {
            key_secondHalf += key_56[i];
        }

        string L_key[16], R_key[16];

        L_key[0] = shift_bit(key_firstHalf, num_leftShift[0]);
        R_key[0] = shift_bit(key_secondHalf, num_leftShift[0]);

        for (int i = 1; i < 16; i++)
        {
            L_key[i] = shift_bit(L_key[i - 1], num_leftShift[i]);
            R_key[i] = shift_bit(R_key[i - 1], num_leftShift[i]);
        }



        string key_48[16], keys_56[16];

        for (int i = 0; i < 16; i++)
        {
            keys_56[i] = L_key[i] + R_key[i];
        }
        for (int i = 0; i < 16; i++)
        {
            key_48[i] = "";
            for (int j = 0; j < 48; j++)
                key_48[i] += keys_56[i][pc_2[j] - 1];
        }

        for(int i=0;i<16;i++){
            cout<<"round "<< i+1 <<"
key="<<Bin_to_Hex(key_48[i])<<endl;
        }
        cout<<"\nthe encrypted C.T.=";

        string plain_txt_64 = Hex_to_Bin(plain_txt);

        string IP = "";

        for (int i = 0; i < 64; i++)
            IP += plain_txt_64[IP_t[i] - 1];

        string L = "", R = "";

        for (int i = 0; i < 32; i++)
            L += IP[i];

        for (int i = 32; i < 64; i++)
            R += IP[i];

        string L_32[16], R_32[16];
        string R_xor_K[16];
        string R_48[16];
        string S_R[16], s[16][8];
        string s_1[16];
        string P_R[16];

        R_48[0] = "";
        for (int j = 0; j < 48; j++)
            R_48[0] += R[E_t[j] - 1];

        R_xor_K[0] = xor_add(R_48[0], key_48[0]);

        for (int j = 0; j <48; j += 6)
            for (int k = j; k < j + 6; k++)
                s[0][j / 6] += R_xor_K[0][k];
        s_1[0] = "";
        for (int j = 0; j < 8; j++)
            s_1[0] += get_element_from_box(s[0][j], j);

        for (int j = 0; j < 32; j++)
            P_R[0] += s_1[0][P[j] - 1];

        L_32[0] = R;
        R_32[0] = "";
        R_32[0] = xor_add(P_R[0], L);

        for (int i = 1; i < 16; i++)

        {
            L_32[i] = R_32[i - 1];
            R_48[i] = "";
            for (int j = 0; j < 48; j++)
                R_48[i] += R_32[i - 1][E_t[j] - 1];

            R_xor_K[i] = xor_add(R_48[i], key_48[i]);

            for (int j = 0; j <48; j += 6)
                for (int k = j; k < j + 6; k++)
                    s[i][j / 6] += R_xor_K[i][k];

            s_1[i] = "";
            for (int j = 0; j < 8; j++)
                s_1[i] += get_element_from_box(s[i][j], j);

            for (int j = 0; j < 32; j++)
                P_R[i] += s_1[i][P[j] - 1];

            L_32[i] = R_32[i - 1];
            R_32[i] = "";
            R_32[i] = xor_add(P_R[i], L_32[i - 1]);

        }

        string encrypted_bin = "", RL;

        RL = R_32[15] + L_32[15];

        for (int i = 0; i < 64; i++)
            encrypted_bin += RL[P_1[i] - 1];

        cout << Bin_to_Hex(encrypted_bin) << endl;
```

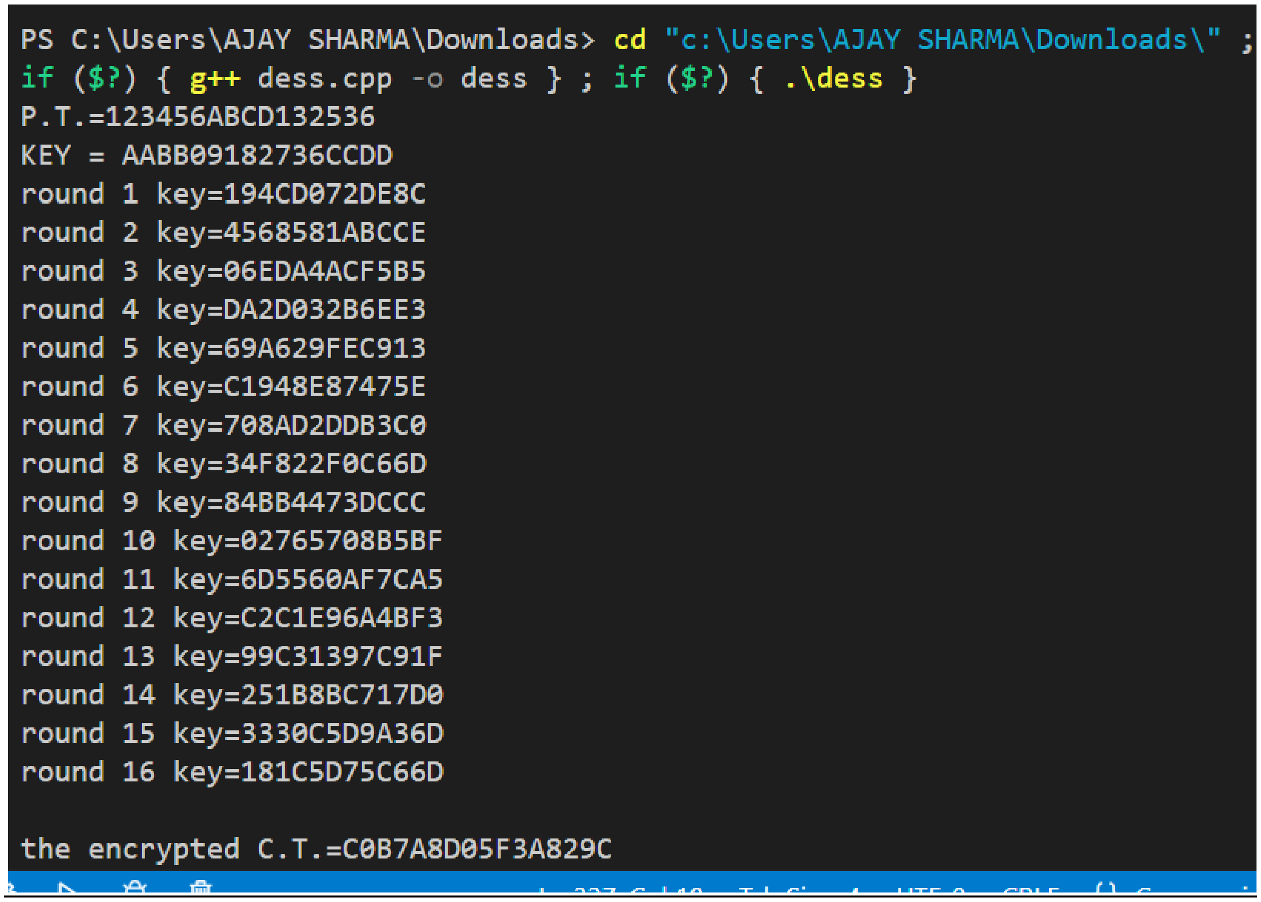
```cpp
    }


};

int main()
{
    DES_Encryption DES;

    bool is_valid;
    string plain_txt, key;

    cout << "P.T.=";

    do {
        is_valid = true;
        cin >> plain_txt;

        if (plain_txt.size() != 16)
            is_valid = false;

        else
        {
            for (int i = 0; i < plain_txt.size(); i++)
                if (!((plain_txt[i] <= 'f' && plain_txt[i] >= 'a') ||
                    (plain_txt[i] <= 'F' && plain_txt[i] >= 'A') ||
                    (plain_txt[i] >= '0' && plain_txt[i] <= '9')))
                {
                    is_valid = false;
                    break;
                }
        }
        if (!is_valid)
            cout << "invalid  ";
    } while (!is_valid);

    cout << "KEY = ";

    do {
        is_valid = true;
        cin >> key;

        if (key.size() != 16)
            is_valid = false;
        else
        {
            for (int i = 0; i < key.size(); i++)
                if (!((key[i] <= 'f' && key[i] >= 'a') ||
                    (key[i] <= 'F' && key[i] >= 'A') ||
                    (key[i] >= '0' && key[i] <= '9')))
                {
                    is_valid = false;
                    break;
                }
        }
        if (!is_valid)
            cout << "invalid input, try again : ";
    } while (!is_valid);

    DES.encrypt(plain_txt, key);
    return 0;
}

string Bin_to_Hex(string s)
{
    string hex = "";
    for (int i = 0; i < s.size(); i += 4)
    {
        string k = "";
        for (int j = i; j < i + 4; j++)
            k += s[j];
        if (k == "0000")
            hex += '0';
        else if (k == "0001")
            hex += '1';
        else if (k == "0010")
            hex += '2';
        else if (k == "0011")
            hex += '3';
        else if (k == "0100")
            hex += '4';
        else if (k == "0101")
            hex += '5';
        else if (k == "0110")
            hex += '6';
        else if (k == "0111")
            hex += '7';
        else if (k == "1000")
            hex += '8';
        else if (k == "1001")
```

```
            hex += '9';
        else if (k == "1010")
            hex += 'A';
        else if (k == "1011")
            hex += 'B';
        else if (k == "1100")
            hex += 'C';
        else if (k == "1101")
            hex += 'D';
        else if (k == "1110")
            hex += 'E';
        else if (k == "1111")
            hex += 'F';
    }
    return hex;
}

string Hex_to_Bin(string s)
{
    string bin = "";
    for (int i = 0; i < s.size(); i++)
    {
        switch (s[i])
        {
        case '0': bin += "0000"; break;
        case '1': bin += "0001"; break;
        case '2': bin += "0010"; break;
        case '3': bin += "0011"; break;
        case '4': bin += "0100"; break;
        case '5': bin += "0101"; break;
        case '6': bin += "0110"; break;
        case '7': bin += "0111"; break;
        case '8': bin += "1000"; break;
        case '9': bin += "1001"; break;
        case 'A':
        case 'a': bin += "1010"; break;
        case 'B':
        case 'b': bin += "1011"; break;
        case 'C':
        case 'c': bin += "1100"; break;
        case 'D':
        case 'd': bin += "1101"; break;
        case 'E':
        case 'e': bin += "1110"; break;
        case 'F':
        case 'f': bin += "1111"; break;
```



```
        }
    }
    return bin;
}

string Dec_to_Bin(int n)
{
    string bin = "";
    while (n > 0)
    {
        bin = (char)(n % 2 + '0') + bin;
        n /= 2;
    }
    while (bin.size() < 4)
        bin = '0' + bin;
    return bin;
}
```

<u>OUTPUT:-</u>



```
PS C:\Users\AJAY SHARMA\Downloads> cd "c:\Users\AJAY SHARMA\Downloads\" ;
if ($?) { g++ dess.cpp -o dess } ; if ($?) { .\dess }
P.T.=123456ABCD132536
KEY = AABB09182736CCDD
round 1 key=194CD072DE8C
round 2 key=4568581ABCCE
round 3 key=06EDA4ACF5B5
round 4 key=DA2D032B6EE3
round 5 key=69A629FEC913
round 6 key=C1948E87475E
round 7 key=708AD2DDB3C0
round 8 key=34F822F0C66D
round 9 key=84BB4473DCCC
round 10 key=02765708B5BF
round 11 key=6D5560AF7CA5
round 12 key=C2C1E96A4BF3
round 13 key=99C31397C91F
round 14 key=251B8BC717D0
round 15 key=3330C5D9A36D
round 16 key=181C5D75C66D

the encrypted C.T.=C0B7A8D05F3A829C
```