LAB:11

NAME: DHRUV PATEL

ROLL NO: CE111

SUBJECT: NIS

Aim: Write a program to implement DES Cipher. Encryption, Decryption

- **Source Code**

```
#include <bits/stdc++.h>
using namespace std;
string hex2bin(string s)
{

    unordered_map<char, string> mp;
    mp['0'] = "0000";
    mp['1'] = "0001";
    mp['2'] = "0010";
    mp['3'] = "0011";
    mp['4'] = "0100";
    mp['5'] = "0101";
    mp['6'] = "0110";
    mp['7'] = "0111";
    mp['8'] = "1000";
    mp['9'] = "1001";
    mp['A'] = "1010";
    mp['B'] = "1011";
    mp['C'] = "1100";
    mp['D'] = "1101";
    mp['E'] = "1110";
    mp['F'] = "1111";
    string bin = "";
    for (int i = 0; i < s.size(); i++) {
        bin += mp[s[i]];
    }
    return bin;
}

string bin2hex(string s)
{
    unordered_map<string, string> mp;
    mp["0000"] = "0";
    mp["0001"] = "1";
    mp["0010"] = "2";
    mp["0011"] = "3";
    mp["0100"] = "4";
    mp["0101"] = "5";
```

```
    mp["0110"] = "6";
    mp["0111"] = "7";
    mp["1000"] = "8";
    mp["1001"] = "9";
    mp["1010"] = "A";
    mp["1011"] = "B";
    mp["1100"] = "C";
    mp["1101"] = "D";
    mp["1110"] = "E";
    mp["1111"] = "F";
    string hex = "";
    for (int i = 0; i < s.length(); i += 4) {
        string ch = "";
        ch += s[i];
        ch += s[i + 1];
        ch += s[i + 2];
        ch += s[i + 3];
        hex += mp[ch];
    }
    return hex;
}

string permute(string k, int* arr, int n)
{
    string per = "";
    for (int i = 0; i < n; i++) {
        per += k[arr[i] - 1];
    }
    return per;
}

string shift_left(string k, int shifts)
{
    string s = "";
    for (int i = 0; i < shifts; i++) {
        for (int j = 1; j < 28; j++) {
            s += k[j];
        }
        s += k[0];
        k = s;
        s = "";
    }
    return k;
}

string xor_(string a, string b)
{
    string ans = "";
```

```
    for (int i = 0; i < a.size(); i++) {
        if (a[i] == b[i]) {
            ans += "0";
        }
        else {
            ans += "1";
        }
    }
    return ans;
}
string encrypt(string pt, vector<string> rkb, vector<string> rk)
{
    pt = hex2bin(pt);
    int initial_perm[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 };
    pt = permute(pt, initial_perm, 64);
    cout << "After initial permutation: " << bin2hex(pt) << endl;

    string left = pt.substr(0, 32);
    string right = pt.substr(32, 32);
    cout << "After splitting: L0=" << bin2hex(left)
        << " R0=" << bin2hex(right) << endl;

    int exp_d[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 }};

    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};

    cout << endl;
    for (int i = 0; i < 16; i++) {
        string right_expanded = permute(right, exp_d, 48);
        string x = xor_(rkb[i], right_expanded);

        string op = "";
        for (int i = 0; i < 8; i++) {
            int row = 2 * int(x[i * 6] - '0') + int(x[i * 6 + 5] - '0');
            int col = 8 * int(x[i * 6 + 1] - '0') + 4 * int(x[i * 6 + 2] -
'0') + 2 * int(x[i * 6 + 3] - '0') + int(x[i * 6 + 4] - '0');
            int val = s[i][row][col];
            op += char(val / 8 + '0');
            val = val % 8;
            op += char(val / 4 + '0');
            val = val % 4;
            op += char(val / 2 + '0');
            val = val % 2;
```

```
            op += char(val + '0');
        }
        op = permute(op, per, 32);
        x = xor_(op, left);
        left = x;
        if (i != 15) {
            swap(left, right);
        }
        cout << "Round " << i + 1 << " " << bin2hex(left) << " "
            << bin2hex(right) << " " << rk[i] << endl;
    }
    string combine = left + right;
    int final_perm[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 };

    string cipher = bin2hex(permute(combine, final_perm, 64));
    return cipher;
}
int main()
{
    string pt, key;

    pt = "654AB43CD12345CC";
    key = "CCDD52491709AABB";
    cout<<"Plain Text(hexadecimal) :"<<pt<<endl;
    cout<<"Key(hexadecimal) :"<<key<<endl;


    key = hex2bin(key);

    int keyp[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 };

    key = permute(key, keyp, 56);

    int shift_table[16] = { 1, 1, 2, 2,
```

```
                            2, 2, 2, 2,
                            1, 2, 2, 2,
                            2, 2, 2, 1 };

    int key_comp[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 };

    string left = key.substr(0, 28);
    string right = key.substr(28, 28);

    vector<string> rkb;
    vector<string> rk;
    for (int i = 0; i < 16; i++) {
        left = shift_left(left, shift_table[i]);
        right = shift_left(right, shift_table[i]);

        string combine = left + right;

        string RoundKey = permute(combine, key_comp, 48);

        rkb.push_back(RoundKey);
        rk.push_back(bin2hex(RoundKey));
    }

    cout << "\nEncryption:\n\n";
    string cipher = encrypt(pt, rkb, rk);
    cout << "\nCipher Text: " << cipher << endl;

    cout << "\nDecryption\n\n";
    reverse(rkb.begin(), rkb.end());
    reverse(rk.begin(), rk.end());
    string text = encrypt(cipher, rkb, rk);
    cout << "\nPlain Text: " << text << endl;
}
```

- **Input Output**

```
[Running] cd "c:\Users\Dhruv j Patel\Desktop\NIS\g
Plain Text(hexadecimal) :ABCD123456132536
Key(hexadecimal) :182736AABB09CCDD

Encryption:

After initial permutation: 12BCDA6303C903B5
After splitting: L0=12BCDA63 R0=03C903B5

Round 1 03C903B5 E46EAA37 1B254CBCF8F5
Round 2 E46EAA37 F4BED3BA 032CDDEB682D
Round 3 F4BED3BA FCA674DF 5D64A0E27B9A
Round 4 FCA674DF 7DA7F9CF D28DA8B5133F
Round 5 7DA7F9CF BD3F1D78 D8A207D71AE2
Round 6 BD3F1D78 D349C586 219E0E54AB7D
Round 7 D349C586 7012AC54 6030E633BCDC
Round 8 7012AC54 EF32A0B8 B0CC7069B5B3
Round 9 EF32A0B8 F4768334 C4D852BEF3A9
Round 10 F4768334 20CD9E4F 26E366325F63
Round 11 20CD9E4F 3CB1C749 AA5503DEA932
Round 12 3CB1C749 7FEB45A2 690379E56F58
Round 13 7FEB45A2 DE7EC87B 85D09979B25A
Round 14 DE7EC87B 0BFA20E2 170BD2F5D42E
Round 15 0BFA20E2 53FBB94E 3E78810C3EEE
Round 16 A6628377 53FBB94E 0D39055875B1

Cipher Text: ADF7432AA979B36C
```

```
Decryption

After initial permutation: A662837753FBB94E
After splitting: L0=A6628377 R0=53FBB94E

Round 1 53FBB94E 0BFA20E2 0D39055875B1
Round 2 0BFA20E2 DE7EC87B 3E78810C3EEE
Round 3 DE7EC87B 7FEB45A2 170BD2F5D42E
Round 4 7FEB45A2 3CB1C749 85D09979B25A
Round 5 3CB1C749 20CD9E4F 690379E56F58
Round 6 20CD9E4F F4768334 AA5503DEA932
Round 7 F4768334 EF32A0B8 26E366325F63
Round 8 EF32A0B8 7012AC54 C4D852BEF3A9
Round 9 7012AC54 D349C586 B0CC7069B5B3
Round 10 D349C586 BD3F1D78 6030E633BCDC
Round 11 BD3F1D78 7DA7F9CF 219E0E54AB7D
Round 12 7DA7F9CF FCA674DF D8A207D71AE2
Round 13 FCA674DF F4BED3BA D28DA8B5133F
Round 14 F4BED3BA E46EAA37 5D64A0E27B9A
Round 15 E46EAA37 03C903B5 032CDDEB682D
Round 16 12BCDA63 03C903B5 1B254CBCF8F5

Plain Text: ABCD123456132536
```

```
[Running] cd "c:\Users\Dhruv j Patel\Desktop\NIS
Plain Text(hexadecimal) :654AB43CD12345CC
Key(hexadecimal) :CCDD52491709AABB

Encryption:

After initial permutation: D31CCD71942D8A22
After splitting: L0=D31CCD71 R0=942D8A22

Round 1 942D8A22 48F767E5 D9C23A20FB7A
Round 2 48F767E5 94F130CE A5CAAACE4D59
Round 3 94F130CE 0EBB0FAC B233828BD35C
Round 4 0EBB0FAC BDA7487C 381E51D1D7A0
Round 5 BDA7487C 0CE4423B 45705CD80E2D
Round 6 0CE4423B F482BCCD 06C5D4DA7A9C
Round 7 F482BCCD 25DEEA3E 5E49233173B9
Round 8 25DEEA3E A40DFF06 ABA129B33823
Round 9 A40DFF06 63EB6CE8 8BB08F3B43D7
Round 10 63EB6CE8 C23162A7 3906CA97C183
Round 11 C23162A7 3A38B6C0 7058A8C62745
Round 12 3A38B6C0 8B8AAA32 90A954FAA3CC
Round 13 8B8AAA32 BC26A496 046E1770D78B
Round 14 BC26A496 FC4AAED3 6735245E342B
Round 15 FC4AAED3 2DFF44BD CA84E1EE7968
Round 16 FAD7CCC4 2DFF44BD C980734FCCA3

Cipher Text: B270BFE672E27D77
```

```
Decryption

After initial permutation: FAD7CCC42DFF44BD
After splitting: L0=FAD7CCC4 R0=2DFF44BD

Round 1 2DFF44BD FC4AAED3 C980734FCCA3
Round 2 FC4AAED3 BC26A496 CA84E1EE7968
Round 3 BC26A496 8B8AAA32 6735245E342B
Round 4 8B8AAA32 3A38B6C0 046E1770D78B
Round 5 3A38B6C0 C23162A7 90A954FAA3CC
Round 6 C23162A7 63EB6CE8 7058A8C62745
Round 7 63EB6CE8 A40DFF06 3906CA97C183
Round 8 A40DFF06 25DEEA3E 8BB08F3B43D7
Round 9 25DEEA3E F482BCCD ABA129B33823
Round 10 F482BCCD 0CE4423B 5E49233173B9
Round 11 0CE4423B BDA7487C 06C5D4DA7A9C
Round 12 BDA7487C 0EBB0FAC 45705CD80E2D
Round 13 0EBB0FAC 94F130CE 381E51D1D7A0
Round 14 94F130CE 48F767E5 B233828BD35C
Round 15 48F767E5 942D8A22 A5CAAACE4D59
Round 16 D31CCD71 942D8A22 D9C23A20FB7A

Plain Text: 654AB43CD12345CC
```