AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course Code: CSE 4174
Course Title: Cyber Security Lab
Academic Semester: Fall 2023

Assignment Topic: Data Encryption Standard (DES)

Submitted on: 22/05/2024

Submitted by
Name: Nafisa Tasnim Neha
Student ID: 20200204020
Lab Section: CSGr-1(A2)

<u>Questions:</u>
Data Encryption Standard (DES) is a symmetric key encryption approach. It has several modes. Two such modes are ECB (Electronic Code Book) and CBC(Cipher Block Chaining).
a. Between ECB and CBC modes, which mode do you think is more secure? Justify your answer with proper explanation.
b. Write a program in C/C++that takes a plaintext and a key as inputs and performs encryption and decryption with the DES mode of your answer from question a.

<u>Solve:</u>
a.
Between ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes in Data Encryption Standard (DES), CBC is generally considered more secure. Here's a brief explanation of each mode and why CBC is preferred in terms of security:

ECB (Electronic Code Book): - In ECB mode, each block plaintext is independently encrypted using the same key. This means that identical blocks of plaintext will produce identical blocks of ciphertext. - The main weakness of ECB is that it does not provide diffusion, i.e., the same plaintext block will always result in the same ciphertext block. This lack of diffusion makes it vulnerable to certain types of attacks.

CBC (Cipher Block Chaining):- In CBC mode, each block of plaintext is XORed with the previous ciphertext block before encryption. This introduces an element of feedback and ensures that identical plaintext blocks do not result in identical ciphertext blocks.- CBC mode provides better diffusion and helps in mitigating certain types of attacks that can exploit patterns in the data.- The chaining of blocks makes it more resistant to frequency analysis and other known-plaintext attacks.

Reasons CBC is considered more secure:
Diffusion: CBC provides better diffusion of the plaintext, meaning that changes in one part of the plaintext affect the entire block of ciphertext.
Error Propagation: In CBC, if an error occurs in one block of ciphertext, it affects the decryption of subsequent blocks, limiting the impact of errors.
Randomization: CBC introduces an element of randomness due to the XOR operation with the previous block, making it harder for attackers to predict or analyze patterns.
Security against certain attacks: CBC is more resistant to certain types of attacks, such as watermarking attacks, where an attacker might attempt to manipulate specific blocks of data.

b.

Code:

```
#include <bits/stdc++.h>
using namespace std;
#include <iostream>
#include <bitset>
#include <cstring>

int arrayresult[64];
int arrayresult2[64];


int initialpermutation[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
};

int inverseinitialpermutation[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
};
```

```
int pc1[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 pc2[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 numberOfShifts[16] = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
int ebitselection[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 permutationp[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
};
int s1Box[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}
};
int s2Box[4][16] =
{
   {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}
};
int s3Box[4][16] =
{
   {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}
};
int s4Box[4][16] =
{
   {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}
};
int s5Box[4][16] =
{
    {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}
};
int s6Box[4][16] =
{
    {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}
};
int s7Box[4][16] =
{
    {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}
};
int s8Box[4][16] =
{
    {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 initialkey[64] =
{
    0, 0, 1, 1, 0, 1, 0, 0,
    0, 0, 1, 0, 1, 1, 0, 1,
    1, 0, 1, 1, 0, 1, 0, 1,
    1, 0, 1, 0, 1, 0, 0, 0,
    0, 0, 0, 1, 1, 1, 0, 1,
    1, 1, 0, 1, 1, 0, 1, 1,
    1, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 1, 0, 0
};
int initialvector[64] =
{
    1, 0, 1, 1, 1, 1, 0, 0,
    1, 1, 1, 0, 1, 0, 1, 1,
    0, 1, 0, 0, 0, 1, 0, 0,
    1, 1, 0, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 0, 0, 0, 1,
    0, 1, 0, 0, 1, 1, 0, 1,
    0, 0, 1, 1, 1, 0, 0, 0,
    1, 1, 0, 0, 1, 0, 0, 1
};


const int ROWS = 8;
const int COLS = 8;
int inputarray[64];

void textToBinaryArray(const char* inputText, bitset<COLS> binaryArray[ROWS])
{
    int len = strlen(inputText);
    for (int i = 0; i < len; i++)
    {

        int asciiValue = static_cast<int>(inputText[i]);
        bitset<8> binaryRepresentation(asciiValue);


        for (int j = 0; j < COLS; j++)
        {
            binaryArray[i][COLS - 1 - j] = binaryRepresentation[j];
        }
    }


    for (int i = len; i < ROWS; i++)
    {
        binaryArray[i] = bitset<COLS>(string("00000001"));
    }

    int demo=0;
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            inputarray[demo]=binaryArray[i][j];
            demo++;
        }

    }

    int demo2=0;
    cout << "The input text is converted to binary:" << endl;
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            cout << inputarray[demo2] << " ";
            demo2++;
        }
        cout << endl;
    }
}

int main()
{
    cout << "Enter the input text: ";
    string inputText;
    getline(cin, inputText);
```

```
bitset<COLS> binaryArray[ROWS];


textToBinaryArray(inputText.c_str(), binaryArray);

int array3[56];
for(int i=0; i<56; i++)
{
   int demo=pc1[i]-1;
   array3[i]=initialkey[demo];
}

int arrayy[48];

int key16[16][48];
int array4[56];

int array5[48];

for(int i=0; i<16; i++)
{

   int demo=numberOfShifts[i];
   for(int j=0; j<demo; j++)
   {

      array4[27]=array3[0];
      for(int w=0; w<27; w++)
      {
         array4[w]=array3[w+1];
      }

      array4[55]=array3[28];
      for(int k=28; k<55; k++)
      {
         array4[k]=array3[k+1];
      }
```

```
            for(int x=0; x<56; x++)
            {
                array3[x]=array4[x];
            }

        }

        for(int m=0; m<56; m++)
        {
            int demo=pc2[m]-1;
            key16[i][m]=array4[demo];
        }
    }


    cout<<"All 16 keys for 16 rounds"<<endl;
    int demo2=0;
    for (int i = 0; i < 16; i++)
    {
        cout<<"Key"<<i+1<<" : ";
        for (int j = 0; j <48; j++)
        {
            cout << key16[i][j] << " ";
            demo2++;
        }
        cout << endl;
    }
    cout<<endl;


cout<<"----------------------------------------------encryption------------------------------------"<<e
ndl;

    for(int i=0; i<64; i++)
    {
        if(inputarray[i]==initialvector[i])
    {
        inputarray[i]=0;
    }
    else
    {
        inputarray[i]=1;
    }
}

for(int round=0; round<16; round++)
{

    int array2[64];
    for(int i=0; i<64; i++)
    {
        int demo=initialpermutation[i]-1;
        array2[i]=inputarray[demo];
    }
    int arrayl0[32];
    int arrayr0[32];

    for(int i=0; i<32; i++)
    {
        arrayl0[i]=array2[i];
        arrayr0[i]=array2[32+i];
    }
    int arrayl1[32];
    for(int i=0; i<32; i++)
    {
        arrayl1[i]=arrayr0[i];
    }

    int array6[48];

    for(int m=0; m<48; m++)
    {
```

```
    int demo=ebitselection[m]-1;
    array6[m]=arrayr0[demo];
}

for(int i=0; i<48; i++)
{
    if(array6[i]==key16[round][i])
    {
        array6[i]=0;
    }
    else
    {
        array6[i]=1;
    }
}

int ar2d[8][6];
int demo10=0;
for(int i=0; i<8; i++)
{
    for(int j=0; j<6; j++)
    {
        ar2d[i][j]=array6[demo10];
        demo10++;
    }
}

int roww=0;
int coll=0;
int prer0[8][4];

for(int i=0; i<8; i++)
{
    if((ar2d[i][0]==0 && ar2d[i][5]==0) )
    {
        roww=0;
    }
```

```
else if((ar2d[i][0]==0 && ar2d[i][5]==1) )
{
   roww=1;
}
else if((ar2d[i][0]==1 && ar2d[i][5]==0) )
{
   roww=2;
}
else if((ar2d[i][0]==1 && ar2d[i][5]==1) )
{
   roww=3;
}
if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
{
   coll=0;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
{
   coll=1;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
{
   coll=2;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
{
   coll=3;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
{
   coll=4;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
{
   coll=5;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
```

```
{
   coll=6;
}
else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
{
   coll=7;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
{
   coll=8;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
{
   coll=9;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
{
   coll=10;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
{
   coll=11;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
{
   coll=12;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
{
   coll=13;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
{
   coll=14;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
{
   coll=15;
}
int value=0;
if(i==0)
{
   value= s1Box[roww][coll];
}
else if(i==1)
{
   value= s2Box[roww][coll];
}
else if(i==2)
{
   value= s3Box[roww][coll];
}
else if(i==3)
{
   value= s4Box[roww][coll];
}
else if(i==4)
{
   value= s5Box[roww][coll];
}
else if(i==5)
{
   value= s6Box[roww][coll];
}
else if(i==6)
{
   value= s7Box[roww][coll];
}
else if(i==7)
{
   value= s8Box[roww][coll];
}
if(value==0)
{
    prer0[i][0]=0;
    prer0[i][1]=0;
    prer0[i][2]=0;
    prer0[i][3]=0;
}
else  if(value==1)
{
    prer0[i][0]=0;
    prer0[i][1]=0;
    prer0[i][2]=0;
    prer0[i][3]=1;
}
else  if(value==2)
{
    prer0[i][0]=0;
    prer0[i][1]=0;
    prer0[i][2]=1;
    prer0[i][3]=0;
}
else  if(value==3)
{
    prer0[i][0]=0;
    prer0[i][1]=0;
    prer0[i][2]=1;
    prer0[i][3]=1;
}
else  if(value==4)
{
    prer0[i][0]=0;
    prer0[i][1]=1;
    prer0[i][2]=0;
    prer0[i][3]=0;
}
else  if(value==5)
{
    prer0[i][0]=0;
    prer0[i][1]=1;
    prer0[i][2]=0;
    prer0[i][3]=1;
}
else  if(value==6)
{
    prer0[i][0]=0;
    prer0[i][1]=1;
    prer0[i][2]=1;
    prer0[i][3]=0;
}
else  if(value==7)
{
    prer0[i][0]=0;
    prer0[i][1]=1;
    prer0[i][2]=1;
    prer0[i][3]=1;
}
else  if(value==8)
{
    prer0[i][0]=1;
    prer0[i][1]=0;
    prer0[i][2]=0;
    prer0[i][3]=0;
}
else  if(value==9)
{
    prer0[i][0]=1;
    prer0[i][1]=0;
    prer0[i][2]=0;
    prer0[i][3]=1;
}
else  if(value==10)
{
    prer0[i][0]=1;
    prer0[i][1]=0;
    prer0[i][2]=1;
    prer0[i][3]=0;
    }
    else  if(value==11)
    {
        prer0[i][0]=1;
        prer0[i][1]=0;
        prer0[i][2]=1;
        prer0[i][3]=1;
    }
    else  if(value==12)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=0;
        prer0[i][3]=0;
    }
    else  if(value==13)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=0;
        prer0[i][3]=1;
    }
    else  if(value==14)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=1;
        prer0[i][3]=0;
    }
    else  if(value==15)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=1;
        prer0[i][3]=1;
    }
}
```

```
int finalr0[32];
int zz=0;
for (int p=0; p<8; p++)
{
   for (int q=0; q<4; q++)
   {

      finalr0[zz] = prer0[p][q];
      zz++;
   }
}
cout<<endl;

int arrayfinalr02[32];

for(int m=0; m<32; m++)
{

   int demo=permutationp[m]-1;
   arrayfinalr02[m]=finalr0[demo];
}

int arrayr1[32];

for(int i=0; i<32; i++)
{
   if(arrayl0[i]==arrayfinalr02[i])
   {
      arrayr1[i]=0;
   }
   else
   {
      arrayr1[i]=1;
   }
}
```

```
        for(int i=0; i<32; i++)
        {
            arrayresult[i]=arrayr1[i];
            arrayresult[32+i]=arrayl1[i];

        }

        cout<<"result of round :"<< round+1<<endl;

        int demoo=0;
        for(int i=0; i<8; i++)
        {
            for(int j=0; j<8; j++)
            {
                cout<<arrayresult[demoo]<<" ";
                demoo++;
            }
            cout<<endl;
        }

        for(int i=0; i<64; i++)
        {
            array2[i]=arrayresult[i];
        }
    }
    for(int m=0; m<64; m++)
    {

        int demo=inverseinitialpermutation[m]-1;
        arrayresult2[m]=arrayresult[demo];
    }

    cout<<"Final result of encryption:"<<endl;
    int demoo=0;
    for(int i=0; i<8; i++)
    {
        for(int j=0; j<8; j++)
        {
            cout<<arrayresult2[demoo]<<" ";
            demoo++;
        }
        cout<<endl;
    }

cout<<"------------------------------------------------Decryption-----------------------------------"<<e
ndl;

    for(int i=0; i<64; i++)
    {
        inputarray[i]=arrayresult2[i];
    }
    for(int round=0; round<16; round++)
    {
        int array2[64];
        for(int i=0; i<64; i++)
        {
            int demo=initialpermutation[i]-1;
            array2[i]=inputarray[demo];
        }

        int arrayl0[32];
        int arrayr0[32];

        for(int i=0; i<32; i++)
        {
            arrayl0[i]=array2[i];
            arrayr0[i]=array2[32+i];
        }
        int arrayl1[32];
        for(int i=0; i<32; i++)
        {
            arrayl1[i]=arrayr0[i];
        }
```

```
int array6[48];

for(int m=0; m<48; m++)
{

   int demo=ebitselection[m]-1;
   array6[m]=arrayr0[demo];
}


for(int i=0; i<48; i++)
{
   if(array6[i]==key16[round][i])
   {
      array6[i]=0;
   }
   else
   {
      array6[i]=1;
   }
}

int ar2d[8][6];
int demo10=0;
for(int i=0; i<8; i++)
{
   for(int j=0; j<6; j++)
   {
      ar2d[i][j]=array6[demo10];
      demo10++;
   }
}

int roww=0;
int coll=0;
int prer0[8][4];
```

```
for(int i=0; i<8; i++)
{
   if((ar2d[i][0]==0 && ar2d[i][5]==0) )
   {
      roww=0;
   }
   else if((ar2d[i][0]==0 && ar2d[i][5]==1) )
   {
      roww=1;
   }
   else if((ar2d[i][0]==1 && ar2d[i][5]==0) )
   {
      roww=2;
   }
   else if((ar2d[i][0]==1 && ar2d[i][5]==1) )
   {
      roww=3;
   }

   if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
   {
      coll=0;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
   {
      coll=1;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
   {
      coll=2;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
   {
      coll=3;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
   {
      coll=4;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
   {
      coll=5;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
   {
      coll=6;
   }
   else if((ar2d[i][1]==0) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
   {
      coll=7;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
   {
      coll=8;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
   {
      coll=9;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
   {
      coll=10;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==0) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
   {
      coll=11;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==0)   )
   {
      coll=12;
   }
   else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==0) && (ar2d[i][4]==1)   )
   {
      coll=13;
```

```
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==0)   )
{
   coll=14;
}
else if((ar2d[i][1]==1) && (ar2d[i][2]==1) && (ar2d[i][3]==1) && (ar2d[i][4]==1)   )
{
   coll=15;
}
int value=0;
if(i==0)
{
   value= s1Box[roww][coll];


}
else if(i==1)
{
   value= s2Box[roww][coll];


}
else if(i==2)
{
   value= s3Box[roww][coll];


}
else if(i==3)
{
   value= s4Box[roww][coll];


}
else if(i==4)
{
   value= s5Box[roww][coll];
```

```
}
else if(i==5)
{
   value= s6Box[roww][coll];

}
else if(i==6)
{
   value= s7Box[roww][coll];

}
else if(i==7)
{
   value= s8Box[roww][coll];

}
if(value==0)
{
   prer0[i][0]=0;
   prer0[i][1]=0;
   prer0[i][2]=0;
   prer0[i][3]=0;
}
else  if(value==1)
{
   prer0[i][0]=0;
   prer0[i][1]=0;
   prer0[i][2]=0;
   prer0[i][3]=1;
}
else  if(value==2)
{
   prer0[i][0]=0;
   prer0[i][1]=0;
   prer0[i][2]=1;
   prer0[i][3]=0;
```

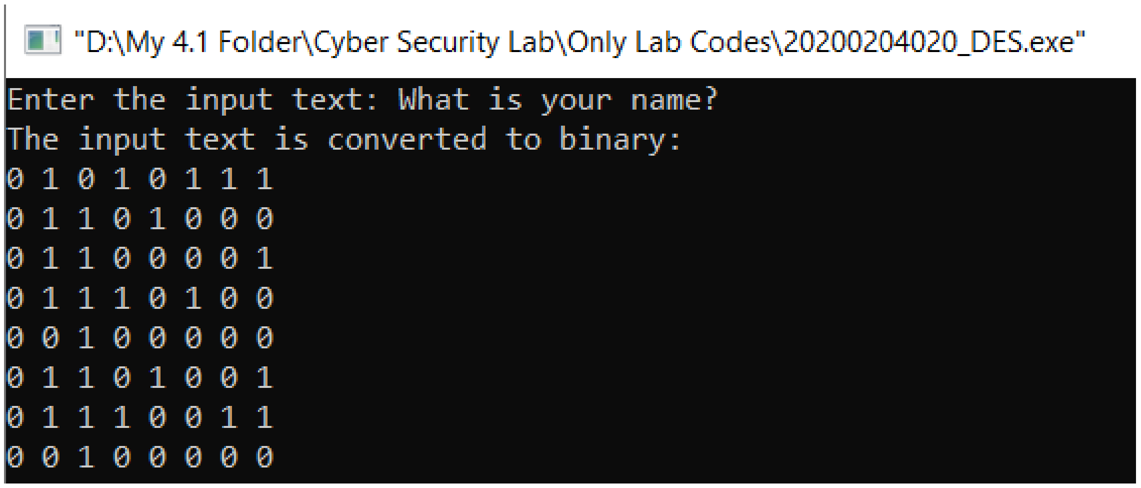
```
}
else  if(value==3)
{
   prer0[i][0]=0;
   prer0[i][1]=0;
   prer0[i][2]=1;
   prer0[i][3]=1;
}
else  if(value==4)
{
   prer0[i][0]=0;
   prer0[i][1]=1;
   prer0[i][2]=0;
   prer0[i][3]=0;
}
else  if(value==5)
{
   prer0[i][0]=0;
   prer0[i][1]=1;
   prer0[i][2]=0;
   prer0[i][3]=1;
}
else  if(value==6)
{
   prer0[i][0]=0;
   prer0[i][1]=1;
   prer0[i][2]=1;
   prer0[i][3]=0;
}
else  if(value==7)
{
   prer0[i][0]=0;
   prer0[i][1]=1;
   prer0[i][2]=1;
   prer0[i][3]=1;
}
else  if(value==8)
```

```
{
   prer0[i][0]=1;
   prer0[i][1]=0;
   prer0[i][2]=0;
   prer0[i][3]=0;
}
else  if(value==9)
{
   prer0[i][0]=1;
   prer0[i][1]=0;
   prer0[i][2]=0;
   prer0[i][3]=1;
}
else  if(value==10)
{
   prer0[i][0]=1;
   prer0[i][1]=0;
   prer0[i][2]=1;
   prer0[i][3]=0;
}
else  if(value==11)
{
   prer0[i][0]=1;
   prer0[i][1]=0;
   prer0[i][2]=1;
   prer0[i][3]=1;
}
else  if(value==12)
{
   prer0[i][0]=1;
   prer0[i][1]=1;
   prer0[i][2]=0;
   prer0[i][3]=0;
}
else  if(value==13)
{
   prer0[i][0]=1;
```

```
        prer0[i][1]=1;
        prer0[i][2]=0;
        prer0[i][3]=1;
    }
    else  if(value==14)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=1;
        prer0[i][3]=0;
    }
    else  if(value==15)
    {
        prer0[i][0]=1;
        prer0[i][1]=1;
        prer0[i][2]=1;
        prer0[i][3]=1;
    }

}
int finalr0[32];
int zz=0;
for (int p=0; p<8; p++)
{
    for (int q=0; q<4; q++)
    {

        finalr0[zz] = prer0[p][q];
        zz++;
    }
}
cout<<endl;

int arrayfinalr02[32];

for(int m=0; m<32; m++)
{
    int demo=permutationp[m]-1;
    arrayfinalr02[m]=finalr0[demo];
  }

  int arrayr1[32];

  for(int i=0; i<32; i++)
  {
    if(arrayl0[i]==arrayfinalr02[i])
    {
      arrayr1[i]=0;
    }
    else
    {
      arrayr1[i]=1;
    }
  }


 for(int i=0; i<32; i++)
  {
    arrayresult[i]=arrayr1[i];
    arrayresult[32+i]=arrayl1[i];

  }
  cout<<"result of round :"<< round+1<<endl;

  int demoo=0;
  for(int i=0; i<8; i++)
  {
    for(int j=0; j<8; j++)
    {
      cout<<arrayresult[demoo]<<" ";
      demoo++;
    }
    cout<<endl;
    }
    for(int i=0; i<64; i++)
    {
        array2[i]=arrayresult[i];
    }
}

for(int m=0; m<64; m++)
{

    int demo=inverseinitialpermutation[m]-1;
    arrayresult2[m]=arrayresult[demo];
}

 for(int i=0;i<64;i++)
 {
     if(arrayresult2[i]==initialvector[i])
     {
         arrayresult2[i]=0;
     }
     else
     {
         arrayresult2[i]=1;
     }
 }
cout<<"Final result of depcryption:"<<endl;
int roy=0;
for(int i=0; i<8; i++)
{
    for(int j=0; j<8; j++)
    {
        cout<<arrayresult2[roy]<<" ";
        roy++;
    }
    cout<<endl;
}
```

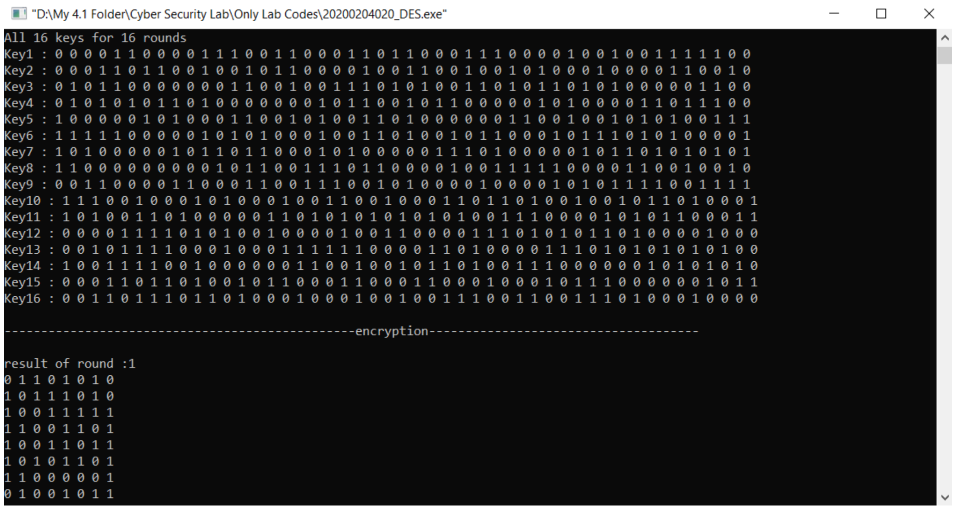
```
    return 0;
}
```

Input:

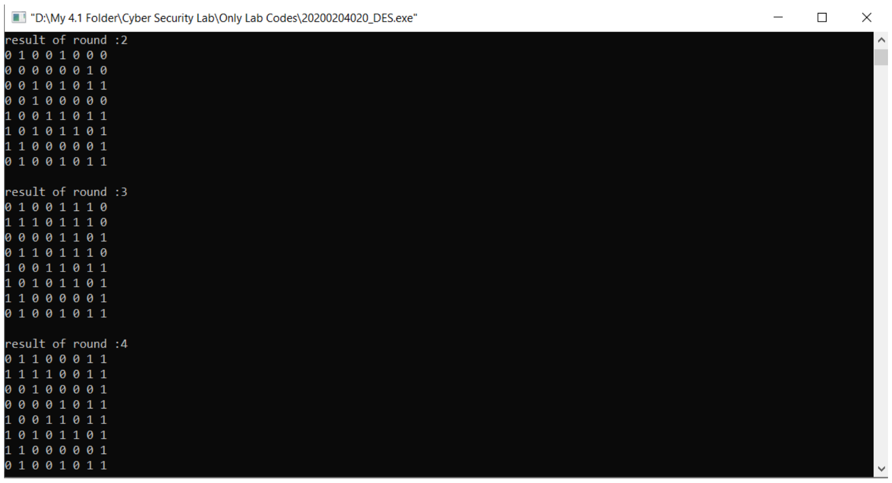


```
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"
Enter the input text: What is your name?
The input text is converted to binary:
0 1 0 1 0 1 1 1
0 1 1 0 1 0 0 0
0 1 1 0 0 0 0 1
0 1 1 1 0 1 0 0
0 0 1 0 0 0 0 0
0 1 1 0 1 0 0 1
0 1 1 1 0 0 1 1
0 0 1 0 0 0 0 0
```

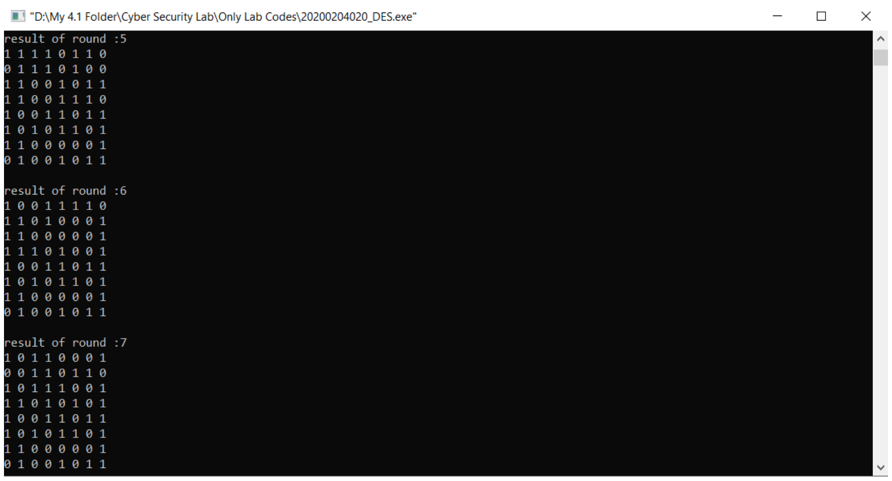
Output:



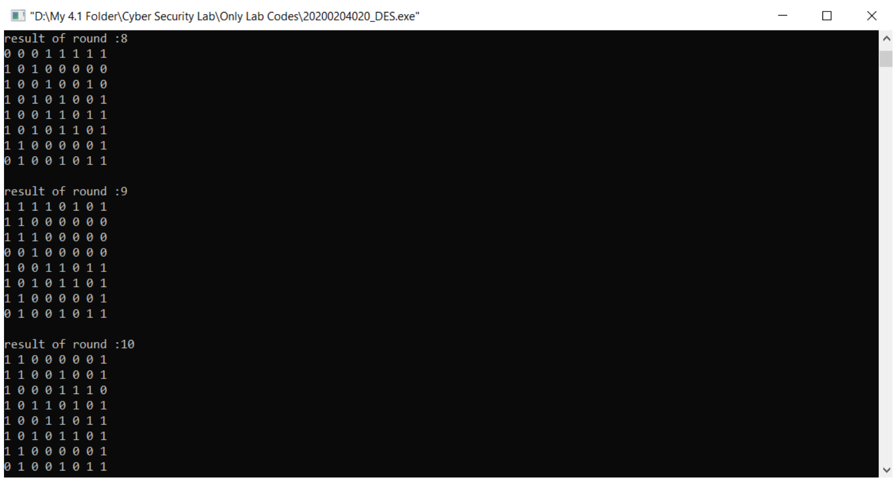
```
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"
All 16 keys for 16 rounds
Key1 : 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0
Key2 : 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0
Key3 : 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0 0
Key4 : 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0
Key5 : 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1
Key6 : 1 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 1
Key7 : 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0 1
Key8 : 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 0
Key9 : 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 1 1 0 0 1 1 1 1
Key10 : 1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 1
Key11 : 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1
Key12 : 0 0 0 0 1 1 1 1 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0
Key13 : 0 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 0
Key14 : 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 1 0 1 0 1 0
Key15 : 0 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 1
Key16 : 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0

-----------------------------------------------encryption-----------------------------------

result of round :1
0 1 1 0 1 0 1 0
1 0 1 1 1 0 1 0
1 0 0 1 1 1 1 1
1 1 0 0 1 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```


"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

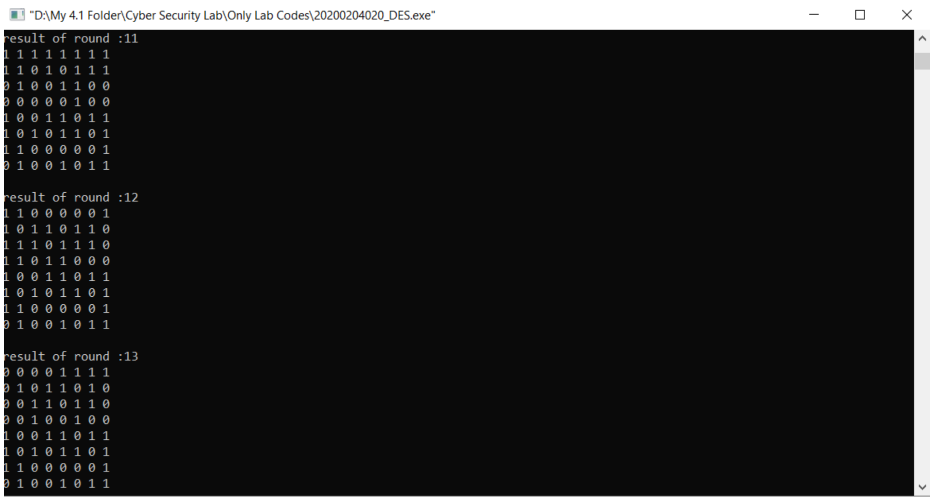
```
result of round :2
0 1 0 0 1 0 0 0
0 0 0 0 0 0 1 0
0 0 1 0 1 0 1 1
0 0 1 0 0 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :3
0 1 0 0 1 1 1 0
1 1 1 0 1 1 1 0
0 0 0 0 1 1 0 1
0 1 1 0 1 1 1 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :4
0 1 1 0 0 0 1 1
1 1 1 1 0 0 1 1
0 0 1 0 0 0 0 1
0 0 0 0 1 0 1 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```


"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :5
1 1 1 1 0 1 1 0
0 1 1 1 0 1 0 0
1 1 0 0 1 0 1 1
1 1 0 0 1 1 1 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :6
1 0 0 1 1 1 1 0
1 1 0 1 0 0 0 1
1 1 0 0 0 0 0 1
1 1 1 0 1 0 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :7
1 0 1 1 0 0 0 1
0 0 1 1 0 1 1 0
1 0 1 1 1 0 0 1
1 1 0 1 0 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```

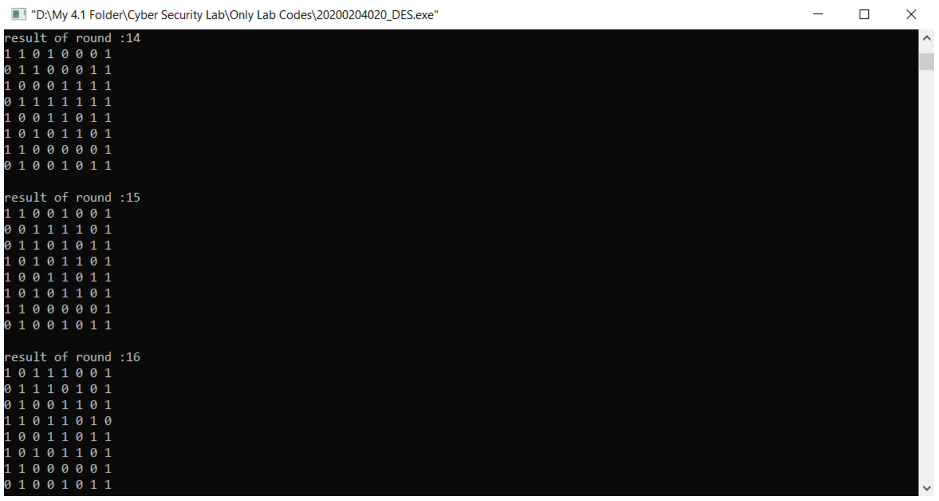

"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :8
0 0 0 1 1 1 1 1
1 0 1 0 0 0 0 0
1 0 0 1 0 0 1 0
1 0 1 0 1 0 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :9
1 1 1 1 0 1 0 1
1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :10
1 1 0 0 0 0 0 1
1 1 0 0 1 0 0 1
1 0 0 0 1 1 1 0
1 0 1 1 0 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```



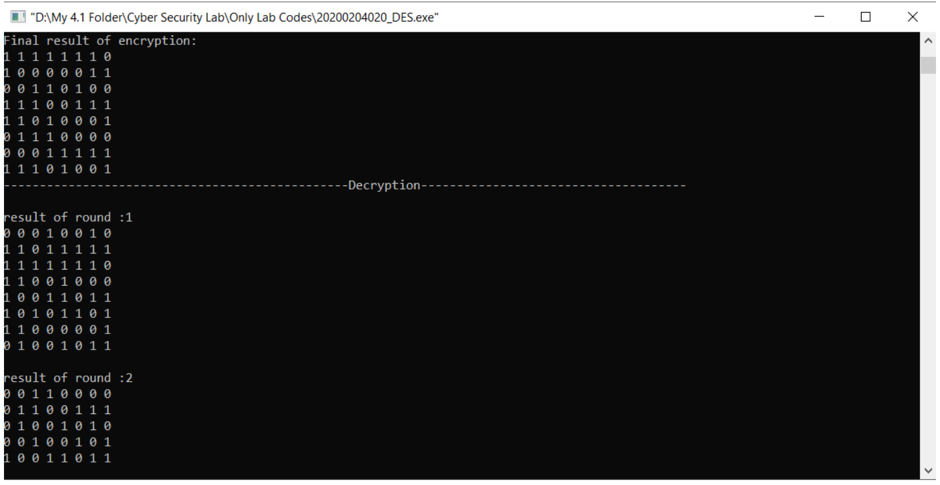
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :11
1 1 1 1 1 1 1 1
1 1 0 1 0 1 1 1
0 1 0 0 1 1 0 0
0 0 0 0 0 1 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :12
1 1 0 0 0 0 0 1
1 0 1 1 0 1 1 0
1 1 1 0 1 1 1 0
1 1 0 1 1 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :13
0 0 0 0 1 1 1 1
0 1 0 1 1 0 1 0
0 0 1 1 0 1 1 0
0 0 1 0 0 1 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```



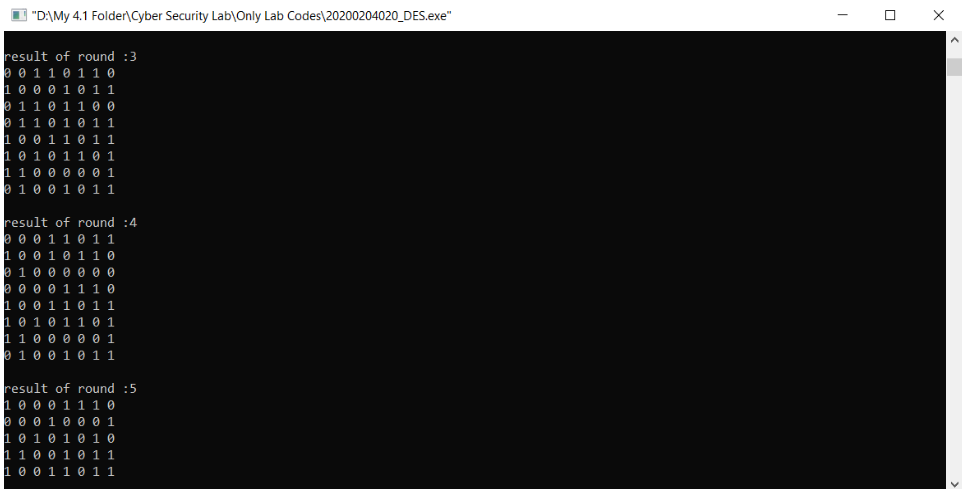
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :14
1 1 0 1 0 0 0 1
0 1 1 0 0 0 1 1
1 0 0 0 1 1 1 1
0 1 1 1 1 1 1 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :15
1 1 0 0 1 0 0 1
0 0 1 1 1 1 0 1
0 1 1 0 1 0 1 1
1 0 1 0 1 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :16
1 0 1 1 1 0 0 1
0 1 1 1 0 1 0 1
0 1 0 0 1 1 0 1
1 1 0 1 1 0 1 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
```



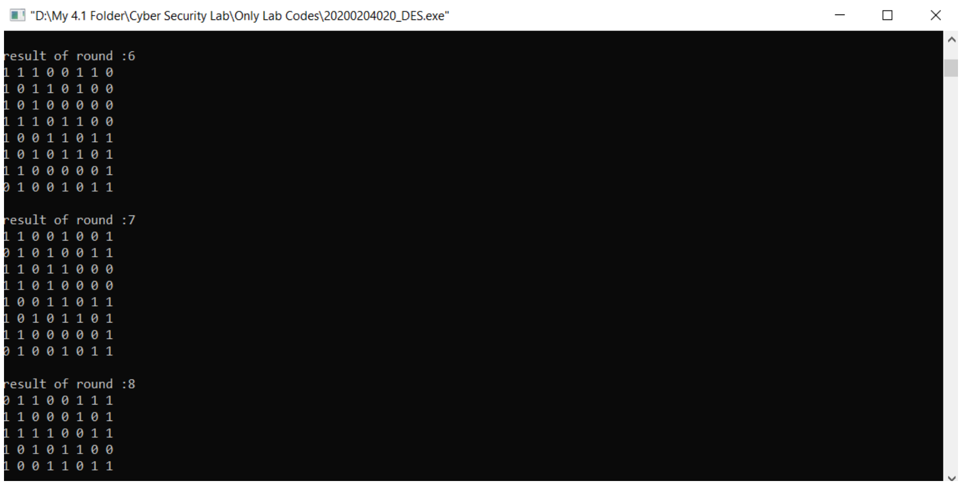
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
Final result of encryption:
1 1 1 1 1 1 1 0
1 0 0 0 0 0 1 1
0 0 1 1 0 1 0 0
1 1 1 0 0 1 1 1
1 1 0 1 0 0 0 1
0 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1
1 1 1 0 1 0 0 1
----------------------------------------------Decryption-----------------------------------

result of round :1
0 0 0 1 0 0 1 0
1 1 0 1 1 1 1 1
1 1 1 1 1 1 1 0
1 1 0 0 1 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :2
0 0 1 1 0 0 0 0
0 1 1 0 0 1 1 1
0 1 0 0 1 0 1 0
0 0 1 0 0 1 0 1
1 0 0 1 1 0 1 1
```



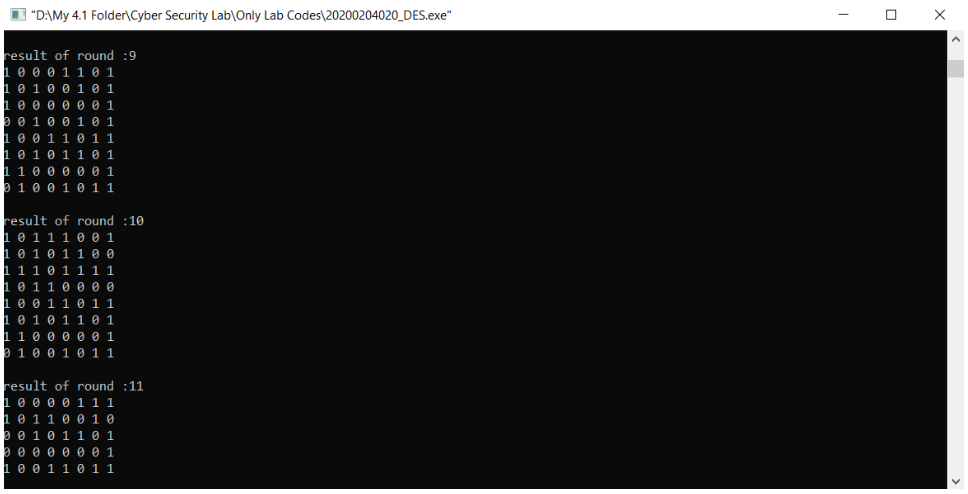
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :3
0 0 1 1 0 1 1 0
1 0 0 0 1 0 1 1
0 1 1 0 1 1 0 0
0 1 1 0 1 0 1 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :4
0 0 0 1 1 0 1 1
1 0 0 1 0 1 1 0
0 1 0 0 0 0 0 0
0 0 0 0 1 1 1 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :5
1 0 0 0 1 1 1 0
0 0 0 1 0 0 0 1
1 0 1 0 1 0 1 0
1 1 0 0 1 0 1 1
1 0 0 1 1 0 1 1
```



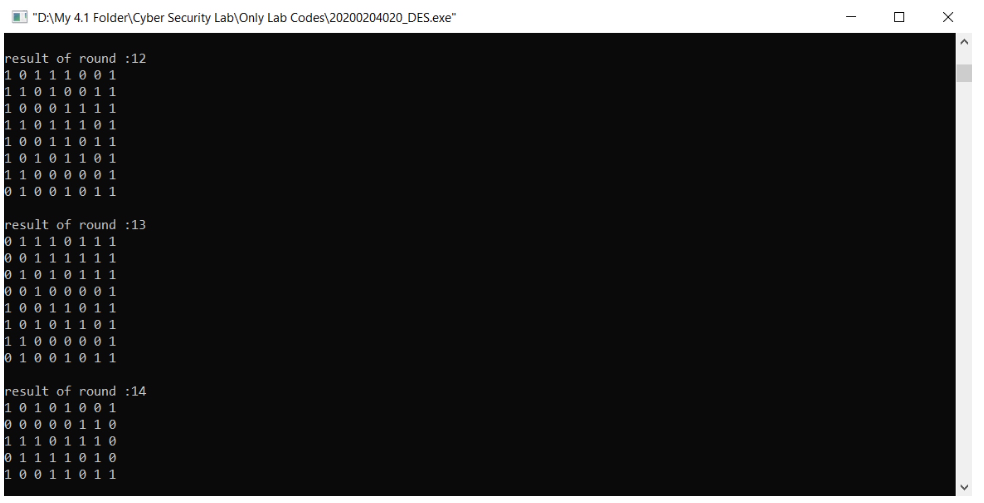
"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :6
1 1 1 0 0 1 1 0
1 0 1 1 0 1 0 0
1 0 1 0 0 0 0 0
1 1 1 0 1 1 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :7
1 1 0 0 1 0 0 1
0 1 0 1 0 0 1 1
1 1 0 1 1 0 0 0
1 1 0 1 0 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :8
0 1 1 0 0 1 1 1
1 1 0 0 0 1 0 1
1 1 1 1 0 0 1 1
1 0 1 0 1 1 0 0
1 0 0 1 1 0 1 1
```

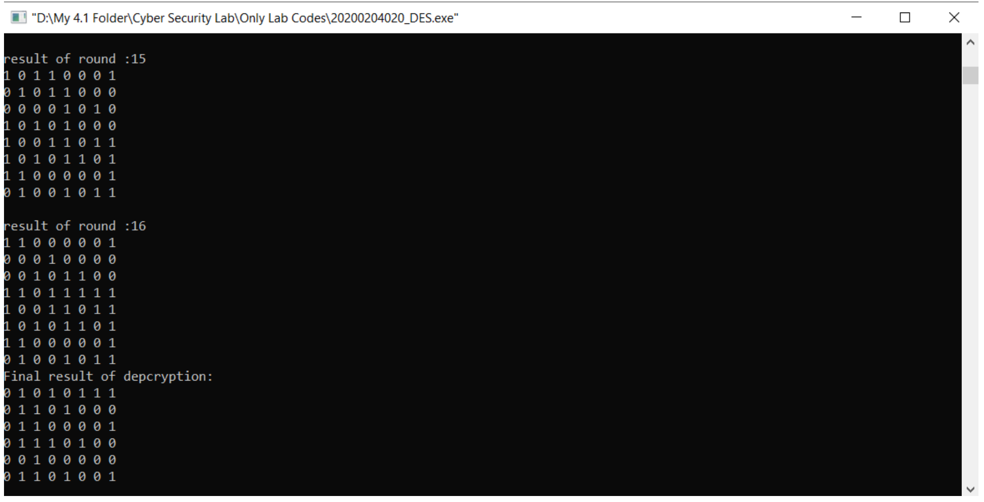


"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :9
1 0 0 0 1 1 0 1
1 0 1 0 0 1 0 1
1 0 0 0 0 0 0 1
0 0 1 0 0 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :10
1 0 1 1 1 0 0 1
1 0 1 0 1 1 0 0
1 1 1 0 1 1 1 1
1 0 1 1 0 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :11
1 0 0 0 0 1 1 1
1 0 1 1 0 0 1 0
0 0 1 0 1 1 0 1
0 0 0 0 0 0 0 1
1 0 0 1 1 0 1 1
```

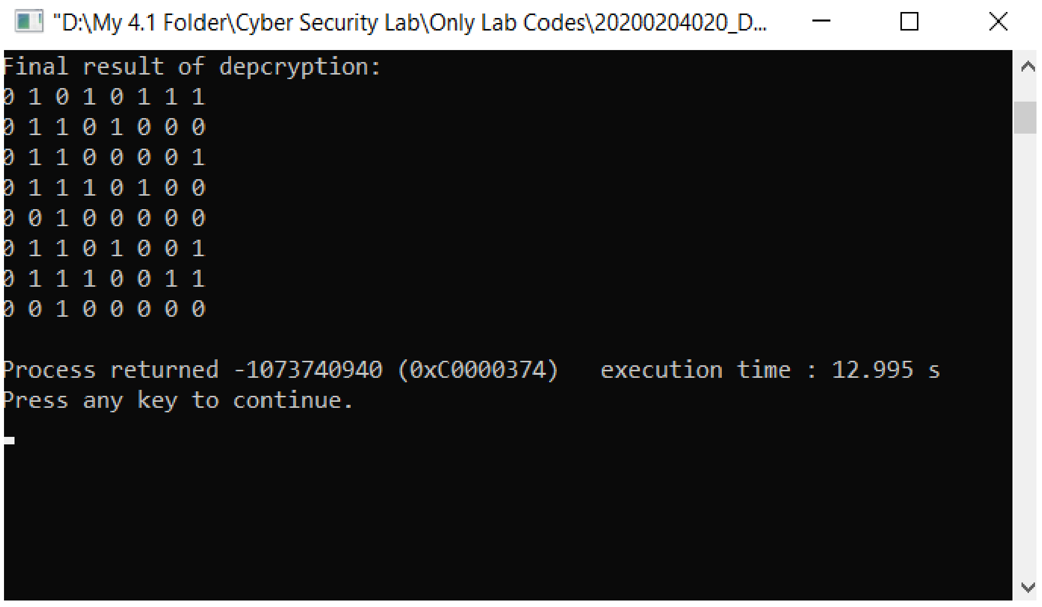

"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :12
1 0 1 1 1 0 0 1
1 1 0 1 0 0 1 1
1 0 0 0 1 1 1 1
1 1 0 1 1 1 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :13
0 1 1 1 0 1 1 1
0 0 1 1 1 1 1 1
0 1 0 1 0 1 1 1
0 0 1 0 0 0 0 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :14
1 0 1 0 1 0 0 1
0 0 0 0 0 1 1 0
1 1 1 0 1 1 1 0
0 1 1 1 1 0 1 0
1 0 0 1 1 0 1 1
```


"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_DES.exe"

```
result of round :15
1 0 1 1 0 0 0 1
0 1 0 1 1 0 0 0
0 0 0 0 1 0 1 0
1 0 1 0 1 0 0 0
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1

result of round :16
1 1 0 0 0 0 0 1
0 0 0 1 0 0 0 0
0 0 1 0 1 1 0 0
1 1 0 1 1 1 1 1
1 0 0 1 1 0 1 1
1 0 1 0 1 1 0 1
1 1 0 0 0 0 0 1
0 1 0 0 1 0 1 1
Final result of depcryption:
0 1 0 1 0 1 1 1
0 1 1 0 1 0 0 0
0 1 1 0 0 0 0 1
0 1 1 1 0 1 0 0
0 0 1 0 0 0 0 0
0 1 1 0 1 0 0 1
```

Final Output:


"D:\My 4.1 Folder\Cyber Security Lab\Only Lab Codes\20200204020_D...

```
Final result of depcryption:
0 1 0 1 0 1 1 1
0 1 1 0 1 0 0 0
0 1 1 0 0 0 0 1
0 1 1 1 0 1 0 0
0 0 1 0 0 0 0 0
0 1 1 0 1 0 0 1
0 1 1 1 0 0 1 1
0 0 1 0 0 0 0 0

Process returned -1073740940 (0xC0000374)   execution time : 12.995 s
Press any key to continue.
```