LAB ASSESSMENT 3

Name: Charan Lalchand Soneji

Registration number: 17BCE2196

Faculty: Prof Marimuthu K.

Course Title: Cryptography fundamentals

Slot: L51+ L52

5. **To implement the Vigenere cipher technique.**
6. **To implement the Simple DES encryption algorithm**

Code

```
#include <bits/stdc++.h>
#include<iostream>
#include<string.h>
#include<string>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
using namespace std;
void vigenere(){
      char s;
    char message[100];
    char key[20];
    cout<<"Enter Message beginning with any random letter to
continue: ";
    cin>>s;
    gets(message);
    cout<<"Enter Key: ";
    gets(key);
      int messageLength = strlen(message), keyLen = strlen(key), i,
j;

    char nKey[messageLength], eMessage[messageLength],
dMessage[messageLength];
    for(i = 0, j = 0; i < messageLength; ++i, ++j){
```

```
        if(j == keyLen)
            j = 0;

        nKey[i] = key[j];
    }
    nKey[i] = '\0';
    for(i = 0; i < messageLength; ++i)
        eMessage[i] = ((message[i] + nKey[i]) % 26) + 'A';
    eMessage[i] = '\0';
    for(i = 0; i < messageLength; ++i)
        dMessage[i] = (((eMessage[i] - nKey[i]) + 26) % 26) + 'A';
    dMessage[i] = '\0';
    cout<<"Original Message: "<<message;
    cout<<"\nKey: "<<key;
    cout<<"\nNew Generated Key: "<<nKey;
    cout<<"\nEncrypted Message: "<<eMessage;
    cout<<"\nDecrypted Message: "<<dMessage;
}
string Permutation(vector<int> array, string inp){
    string out = "";
    for(int i=0;i<array.size();i++)
        out += inp[array[i]-1];
    return out;
}
class S_DES{
public:
    string KEY,K1,K2,IPOut,InvIPOut;
    string F1Out;
    string INPUT,OUTPUT;
    void initialize(string key){
        if(key.size()!=10){
            cout<<"\nInValid Key-Length "<<key<<" "<<key.size();
            exit(1);
        }
```

```
        KEY = key;
         keygen();
      }
    void keygen(){a
        cout<<"Enter P10 permutation array: ";
        vector<int> P10(10,0);
        for(int i=0;i<10;i++)
            cin>>P10[i];
        string P10_output = Permutation(P10,KEY);
        cout<<"P10 output while generating key: "<<P10_output<<endl;
        string P10_left = P10_output.substr(0,5), P10_right =
P10_output.substr(5,5);
        string pl = LShift(P10_left,1), pr = LShift(P10_right,1);
        string plpr = pl+pr;
        cout<<"Enter P8 permutation array: ";
        vector<int> P8(10,0);
        for(int i=0;i<8;i++)
            cin>>P8[i];
        K1 = Permutation(P8,plpr);
        cout<<"K1: "<<K1<<endl;
         string pl1=LShift(pl,2), pr1=LShift(pr,2);
        plpr = pl1+pr1;
        K2 = Permutation(P8,plpr);
        cout<<"K2: "<<K2<<endl;
    }

     string LShift(string input,int n){
        string output = input;
        char firstbit;
         while(n--){
            firstbit = output[0];
            output = output.substr(1,output.size()-1);
            output += firstbit;
        }
```

```
    return output;
}
void DES_Encryption(){
     IP();
    string LIP = IPOut.substr(0,4);
    string RIP = IPOut.substr(4,4);
    cout<<"IP output: "<<IPOut<<endl;
    Function_F(LIP,RIP,1);
    cout<<"Fn Output: "<<F1Out<<endl;
    string L1 = F1Out.substr(0,4), R1 = F1Out.substr(4,4);
     Function_F(R1,L1,2);
    cout<<"Fn Output second time: "<<F1Out<<endl;
    InvIP(F1Out);
    cout<<"Encrypted Cipher-string: "<<InvIPOut<<endl;
 }
 void IP(){
    vector<int> IP_array(8,0);
    cout<<"Enter initial Permutation array: ";
    for(int i=0;i<8;i++)
        cin>>IP_array[i];
    IPOut = Permutation(IP_array,INPUT);
 }
void InvIP(string input){
vector<int> InvIPArray(8,0);
cout<<"Enter Inverse initial Permutation: ";
for(int i=0;i<8;i++)
    cin>>InvIPArray[i];
InvIPOut = Permutation(InvIPArray,input);
}
 void Function_F(string linput,string rinput,int key)
 {
    cout<<"Enter E/P array: ";
    vector<int> E_P(8,0);
    for(int i=0;i<8;i++)
```

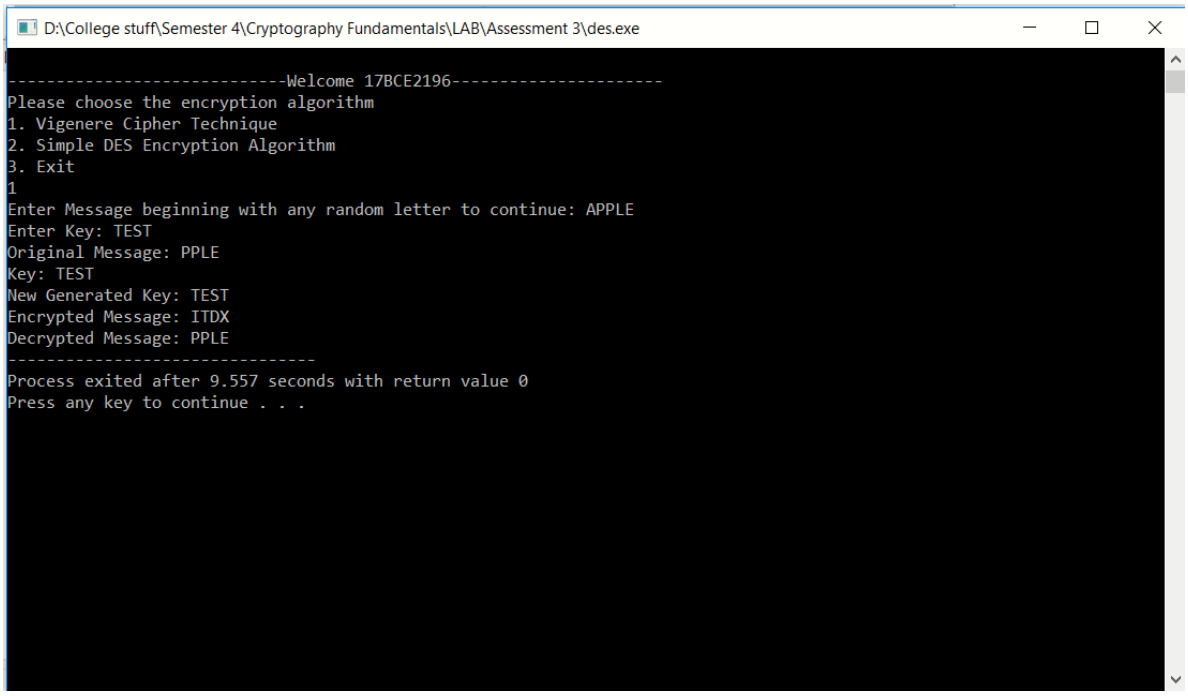
```
            cin>>E_P[i];
        string E_POutput = Permutation(E_P,rinput);
        string EXOR_Output;
        if(key == 1)
            EXOR_Output = EX_OR(E_POutput,K1);
        else
            EXOR_Output = EX_OR(E_POutput,K2);
        string LEXOR = EXOR_Output.substr(0,4),REXOR =
EXOR_Output.substr(4,4);
        string SBOX0_Output=SBOX0(LEXOR);
        string SBOX1_Output=SBOX1(REXOR);
        string SBOX_Output = SBOX0_Output+SBOX1_Output;
        cout<<"Enter P4 Operation array: ";
        vector<int> P4(4,0);
        for(int i=0;i<4;i++)
            cin>>P4[i];
        string P4_Output = Permutation(P4,SBOX_Output);
        string fk_Output = EX_OR(P4_Output,linput);
        F1Out = fk_Output + rinput;
      }
      string EX_OR(string a,string b){
        string output = "";
        for(int i=0;i<a.size();i++){
            if(a[i] == b[i])
                output += "0";
            else
                output += "1";
        }
        return output;
      }
    string SBOX0(string l)
      {
        cout<<"Enter Input for S0\n";
        vector<int> temp(4,0);
```

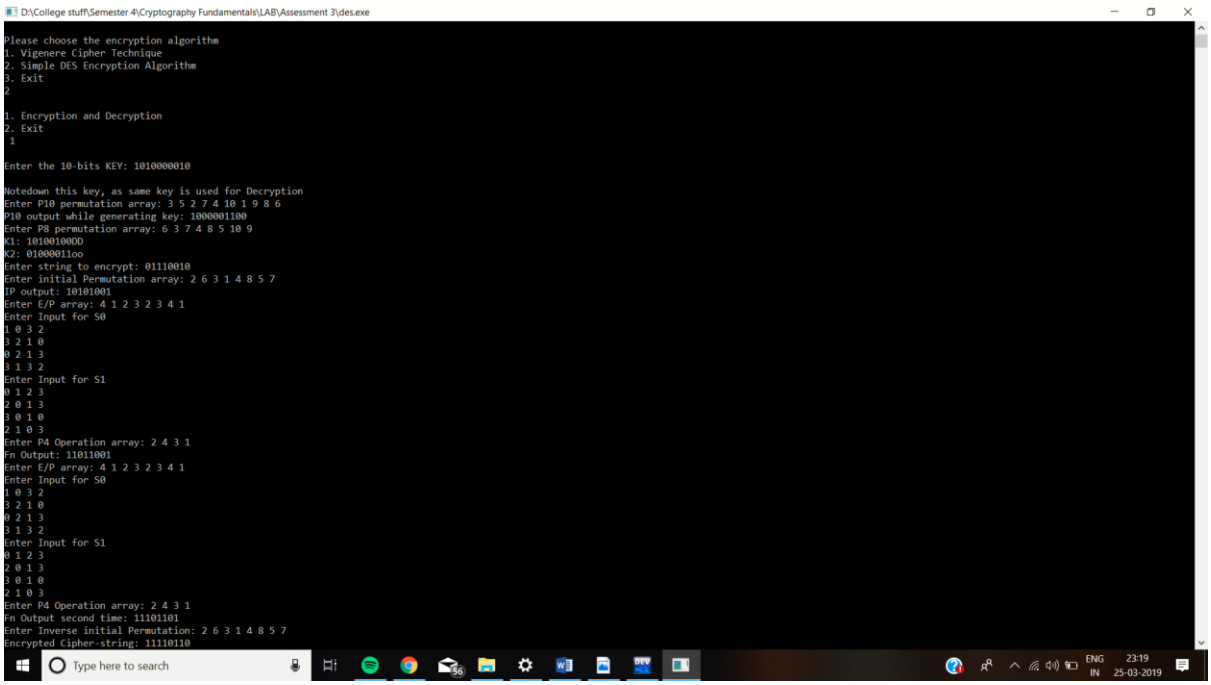
```
        vector<vector<int> > S0(4,temp);
        for(int i=0;i<4;i++){
            for(int j = 0;j<4;j++)
                cin>>S0[i][j];
        }
        string bits[]={"00","01","10","11"};
         string lrow = l.substr(0,1)+l.substr(3,1),lcol =
l.substr(1,1)+l.substr(2,1);
         string SO;
        int i,lr,lc,b;
        for(i=0;i<4;i++){
            if(lrow == bits[i])
                lr=i;
            if(lcol == bits[i])
                lc=i;
        }
        b=S0[lr][lc];
        return bits[b];
     }
    string SBOX1(string l)
     {
        cout<<"Enter Input for S1\n";
        vector<int> temp(4,0);
        vector<vector<int> > S0(4,temp);
        for(int i=0;i<4;i++){
            for(int j = 0;j<4;j++)
                cin>>S0[i][j];
        }
        string bits[]={"00","01","10","11"};
         string lrow = l.substr(0,1)+l.substr(3,1),lcol =
l.substr(1,1)+l.substr(2,1);
         string SO;
        int i,lr,lc,b;
        for(i=0;i<4;i++){
            if(lrow == bits[i])
```

```
                lr=i;
            if(lcol == bits[i])
                lc=i;
        }
        b=S0[lr][lc];
        return bits[b];
      }
};
int main()
{
      cout<<"\n------------------------------Welcome 17BCE2196-------
---------------";
    cout<<"\nPlease choose the encryption algorithm";
    cout<<"\n1. Vigenere Cipher Technique";
    cout<<"\n2. Simple DES Encryption Algorithm";
    cout<<"\n3. Exit\n";
     int o;
    cin>>o;
     switch(o){
           case 1:
                vigenere();
                break;
           case 2:
                int i,n=10,choice;
    string key;
    S_DES S;
    while(1){
        cout<<"\n1. Encryption and Decryption\n2. Exit\n ";
        cin>>choice;
        switch(choice){
            case 1:
            cout<<"\nEnter the 10-bits KEY: ";
            cin>>key;
            cout<<"\nNotedown this key, as same key is used for
Decryption\n";
```

```
            S.initialize(key);
            cout<<"Enter string to encrypt: ";
            cin>>S.INPUT;
            S.DES_Encryption();
         break;
            case 2:
            exit(0);
            default:
              cout<<"\nInvalid option\n";
            break;
        }
    }
      }
    return 0;
}
```

OUTPUT SNIPPETS





TERMINAL OUTPUT FOR DES

```
Please choose the encryption algorithm
1. Vigenere Cipher Technique
2. Simple DES Encryption Algorithm
3. Exit
2

1. Encryption and Decryption
2. Exit
 1

Enter the 10-bits KEY: 1010000010

Notedown this key, as same key is used for Decryption
Enter P10 permutation array: 3 5 2 7 4 10 1 9 8 6
P10 output while generating key: 1000001100
Enter P8 permutation array: 6 3 7 4 8 5 10 9
K1: 10100100DD
K2: 01000011oo
```

```
Enter string to encrypt: 01110010
Enter initial Permutation array: 2 6 3 1 4 8 5 7
IP output: 10101001
Enter E/P array: 4 1 2 3 2 3 4 1
Enter Input for S0
1 0 3 2
3 2 1 0
0 2 1 3
3 1 3 2
Enter Input for S1
0 1 2 3
2 0 1 3
3 0 1 0
2 1 0 3
Enter P4 Operation array: 2 4 3 1
Fn Output: 11011001
Enter E/P array: 4 1 2 3 2 3 4 1
Enter Input for S0
1 0 3 2
3 2 1 0
0 2 1 3
3 1 3 2
Enter Input for S1
0 1 2 3
2 0 1 3
3 0 1 0
2 1 0 3
Enter P4 Operation array: 2 4 3 1
Fn Output second time: 11101101
Enter Inverse initial Permutation: 2 6 3 1 4 8 5 7
Encrypted Cipher-string: 11110110

1. Encryption and Decryption
2. Exit
```