SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Nitin Krishna H	USN:1SI18CS068	Batch No:A4	Date:03/12/2021	
Evaluation:				
Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

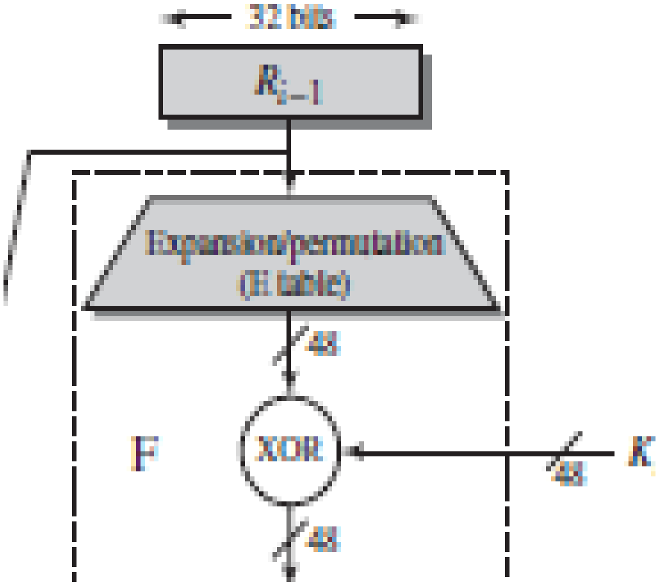
Sl.No	Name of the Faculty In-Charge	Signature
1.	Mr Bhaskar G	
2.	Mrs Thejaswini S	

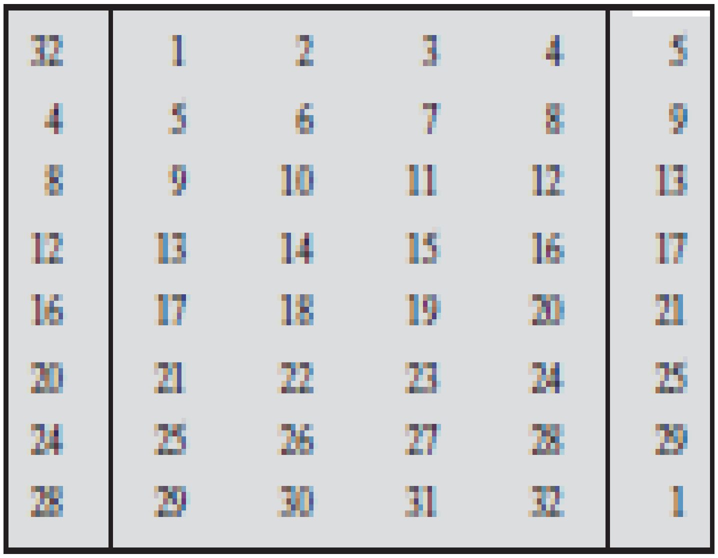
Question No: 6

i) Given 64-bit output of (i-1)th round of DES, 48-bit ith round key K_i and E table, find the 48-bit input for S-box.

ii) Given 48-bit input to S-box and permutation table P, find the 32-bit output R_i of ith round of DES algorithm.

i) **Algorithm:** Follow the flow-chart and tables given below.





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

Figure: Generation of 48-bit input for S-box.

Table: Expansion Permutation

ii) **Algorithm:** The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

48-bit input to S-box

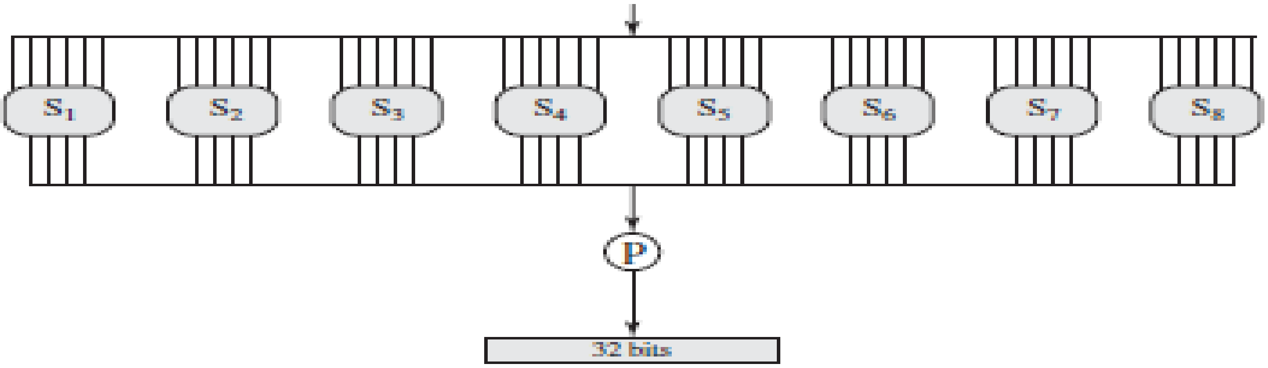
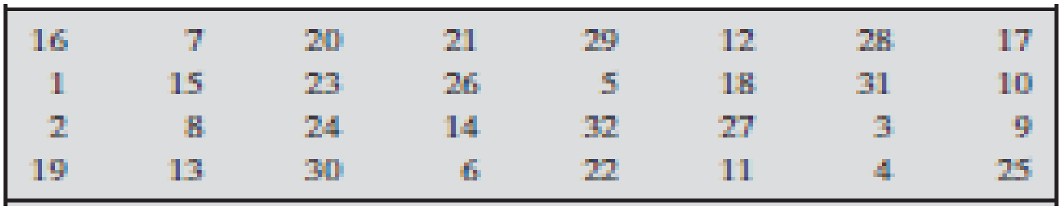
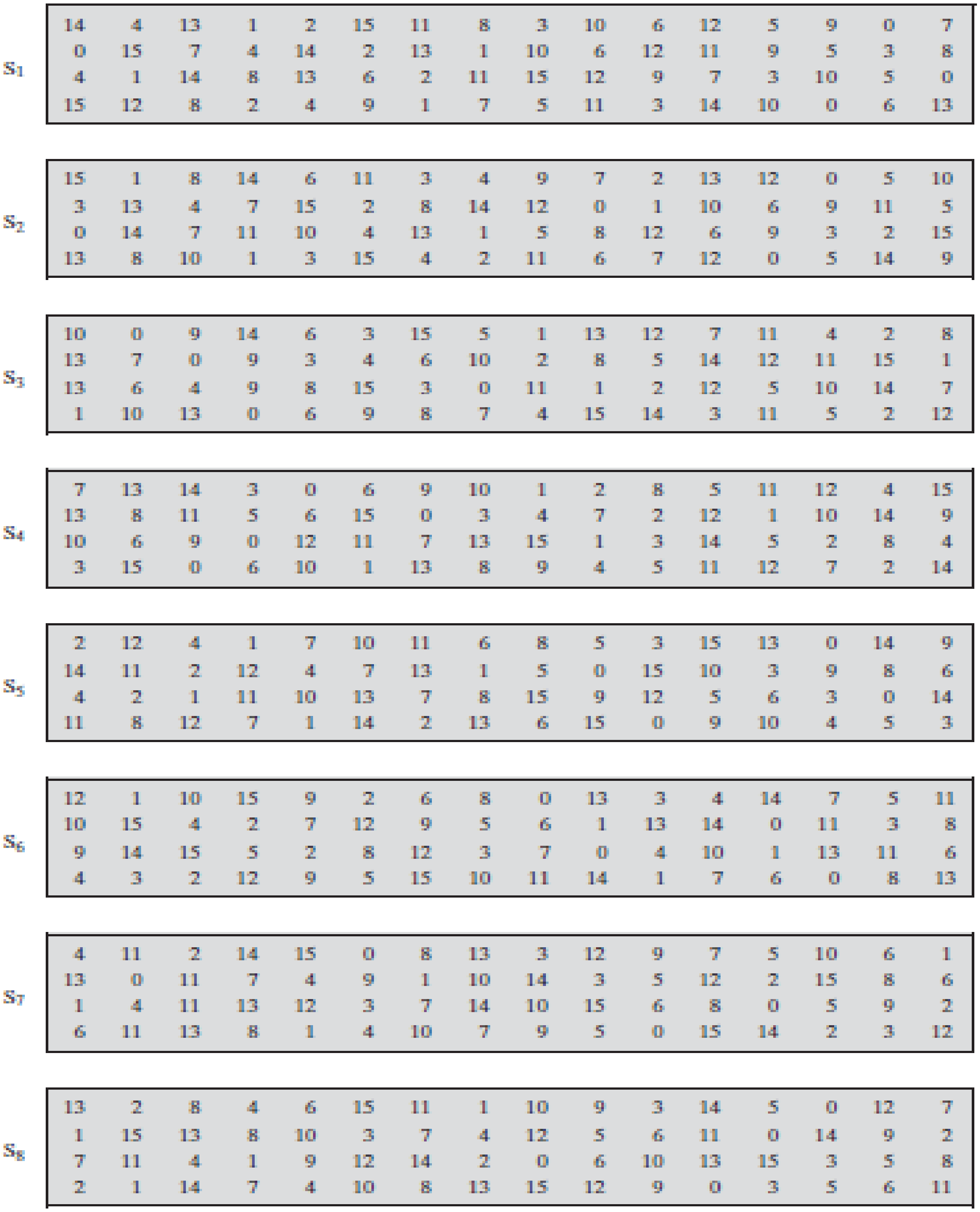


Figure: The 32-bit output R_i of i^{th} round, given 48-bit input



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

Table: Permutation Function (P)



S_1	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

S_2	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

S_3	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

S_4	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

S_5	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

S_6	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

S_7	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

S_8	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

PROGRAM (PART A)

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

//48 bit expansion table consisting of 1 to 32
int E[]={ 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 };

string expansionPermute(string input)
{
            string res="";
            for(int i=0;i<48;i++)
                        res += input[E[i]-1];
            return res;
}

string XOR(string input1,string input2)
{
            string res="";
            for(int i=0;i<input1.length();i++)
                        res += (input1[i]==input2[i])? "0" : "1";//if same then zero else one
    return res;
}

int main()
{
            //get round number [optional]
            /*int     i;
            cout<<"\nEnter Round number(i):";
            cin>>i;*/

            //input (i-1)th round output
            unsigned long long hexInput;
    cout<<"Enter 64-bit(i-1)th round output in hex:";
            cin>>hex>>hexInput;

            //convert to binary
            string input = bitset<64>(hexInput).to_string();
    cout<<"\n64-bitBinaryInput="<<input<<endl;

    // optional if key is taken as input from user
    //get ith round key(48 bit) [can be input from user]
    /*string Ki;
            ifstream fin;
            fin.open("keygen.txt");
            for(int j=1;j<=i;j++)
                        fin>>Ki;
            if(Ki.length()==0)
            {
                        cout<<"\nkeygen.txt not found!!!\n"<<endl;
                        exit(1);
            }*/
            unsigned long long hexkey;
            cout<<"\nEnter 48 bit key in hexadecimal format : ";
            cin>>hex>>hexkey;
            string Ki = bitset<48>(hexkey).to_string();
            cout<<"keyforithround(Ki)="<<Ki<<endl;

            //extract right 32 bits
            string Ri_1 = input.substr(32,32);//32 bit Right half of inputR[i-1]
            cout<<"\nRight half of 64-bit input , Ri_1= "<<Ri_1<<endl;

            //expand right 32 bits
            string R48 = expansionPermute(Ri_1);
            cout<<"Ri_1 after expansion permutation = "<<R48<<endl;

            //XOR with 48 bit key
    string sBoxInput = XOR(R48,Ki);
            cout<<"\nInput to s-box: "<<sBoxInput<<endl<<endl;

    return 0;
}
```

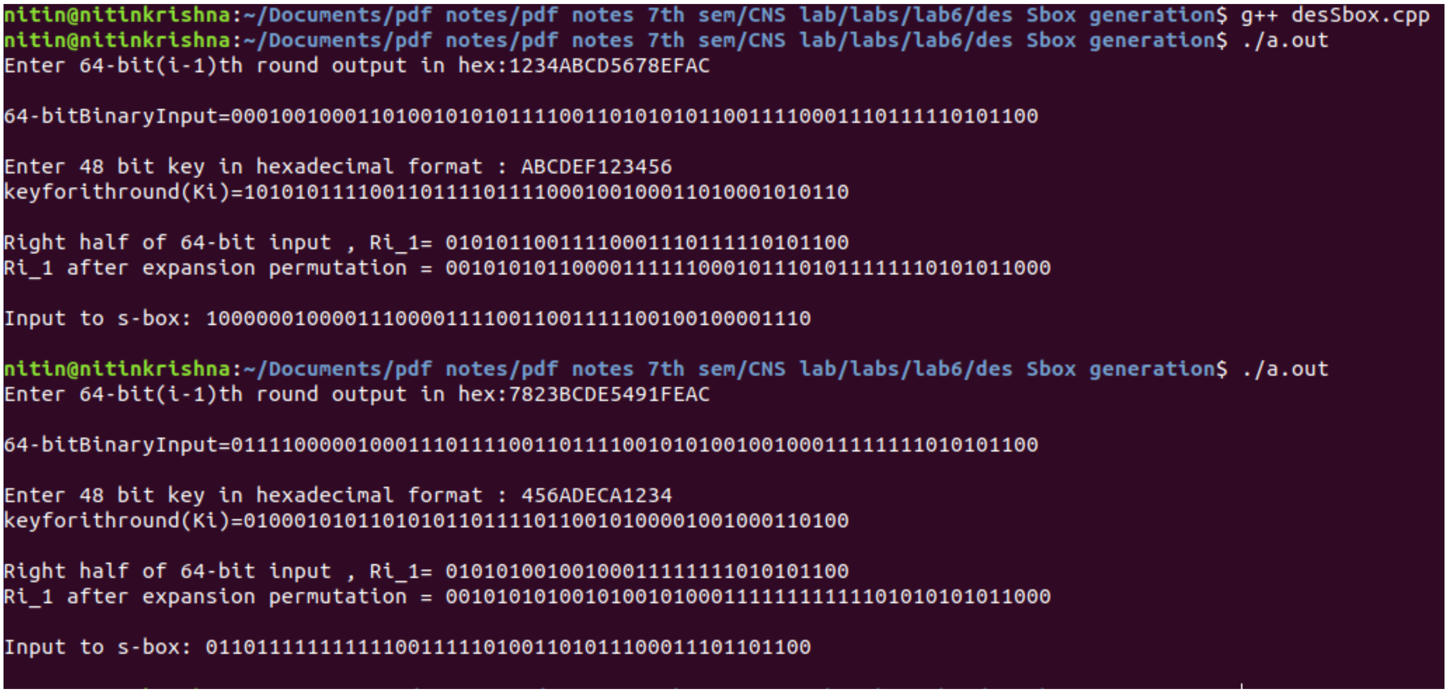
PROGRAM(PART B)

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

unsigned int sBoxes[8][64]={
                                        {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 permTable[]={ 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  };

string substitution(string input)
{
        string res="";//to store final s-boxoutput
        for(int i=0;i<8;i++)//8 sboxes
        {
                string sInput = input.substr(6*i,6);//extract 6 bit input to s box

                //get row and column of sbox
                int row = bitset<2>(sInput.substr(0,1) + sInput.substr(5,1)).to_ulong();
                int col = bitset<4>(sInput.substr(1,4)).to_ulong();

                res += bitset<4>(sBoxes[i][row*16+col]).to_string();
        }
        return res;
}
```

```
string permute(string input)
{
        string res="";
        for(int i=0;i<32;i++)
                res += input[permTable[i]-1];
        return res;
}

string XOR(string input1,string input2)
{
        string res="";
        for(int i=0;i<input1.length();i++)
                res += (input1[i]==input2[i])?"0":"1";
        return res;
}

int main()
{
        //get 64 bit (i-1)th round output
        unsigned long long hexInput;
        cout<<"\nEnter 64-bit (i-1)th round output inhex(16-digits) : ";
        cin>>hex>>hexInput;
        string input = bitset<64>(hexInput).to_string();
        cout<<"Round(i-1) output:"<<input<<endl;

        //extract left 32 bit from 64 bit output
        string Li_1 = input.substr(0,32);
        cout<<"\nLi_1 : "<<Li_1<<endl;

        //get sbox input
        unsigned long long hexSBoxInput;
        cout<<"\nEnter 48-bit input for S-Box inhex(12-digits) : ";
        cin>>hex>>hexSBoxInput;
        string sBoxinput = bitset<48>(hexSBoxInput).to_string();
        cout<<"S-BoxInput : "<<sBoxinput<<endl;

        //calculate sbox output
        string sBoxOutput = substitution(sBoxinput);
        cout<<"\nS-Boxoutput="<<sBoxOutput<<endl;

        //permute S box output
        string P = permute(sBoxOutput);
        cout<<"\nPermuted output="<<P<<endl;

        //xor permuted output and left half
        string Ri = XOR(P,Li_1);
        cout<<"\nOutput of ith round(Ri) = "<<Ri<<endl;

        return 0;
```

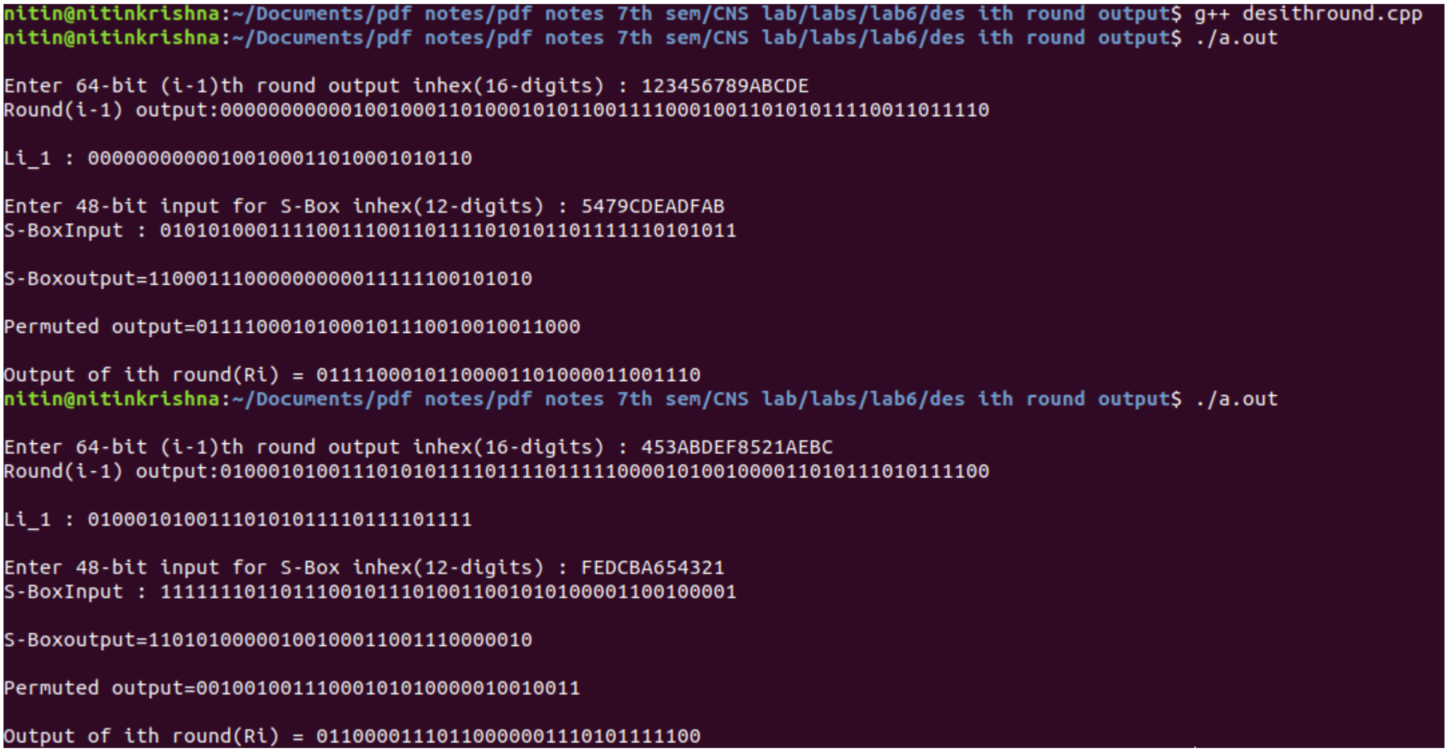
}

OUTPUT(PART A)



```
nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des Sbox generation$ g++ desSbox.cpp
nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des Sbox generation$ ./a.out
Enter 64-bit(i-1)th round output in hex:1234ABCD5678EFAC

64-bitBinaryInput=0001001000110100101010111100110101010110011110001110111110101100

Enter 48 bit key in hexadecimal format : ABCDEF123456
keyforithround(Ki)=101010111100110111101111000100100011010001010110

Right half of 64-bit input , Ri_1= 01010110011110001110111110101100
Ri_1 after expansion permutation = 001010101100001111110001011101011111110101011000

Input to s-box: 100000010000111000011110011001111100100100001110

nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des Sbox generation$ ./a.out
Enter 64-bit(i-1)th round output in hex:7823BCDE5491FEAC

64-bitBinaryInput=0111100000100011101111001101111001010100100100011111111010101100

Enter 48 bit key in hexadecimal format : 456ADECA1234
keyforithround(Ki)=010001010110101011011110110010100001001000110100

Right half of 64-bit input , Ri_1= 01010100100100011111111010101100
Ri_1 after expansion permutation = 001010101001010010100011111111111101010101011000

Input to s-box: 011011111111111001111101001101011100011101101100
```

OUTPUT(PART B)



```
nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des ith round output$ g++ desithround.cpp
nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des ith round output$ ./a.out

Enter 64-bit (i-1)th round output inhex(16-digits) : 123456789ABCDE
Round(i-1) output:0000000000010010001101000101011001111000100110101011110011011110

Li_1 : 00000000000100100011010001010110

Enter 48-bit input for S-Box inhex(12-digits) : 5479CDEADFAB
S-BoxInput : 010101000111100111001101111010101101111110101011

S-Boxoutput=11000111000000000011111100101010

Permuted output=01111000101000101110010010011000

Output of ith round(Ri) = 01111000101100001101000011001110
nitin@nitinkrishna:~/Documents/pdf notes/pdf notes 7th sem/CNS lab/labs/lab6/des ith round output$ ./a.out

Enter 64-bit (i-1)th round output inhex(16-digits) : 453ABDEF8521AEBC
Round(i-1) output:0100010100111010101111011110111110000101001000011010111010111100

Li_1 : 01000101001110101011110111101111

Enter 48-bit input for S-Box inhex(12-digits) : FEDCBA654321
S-BoxInput : 111111101101110010111010011001010100001100100001

S-Boxoutput=11010100000010010001100111000010

Permuted output=00100100111000101010000010010011

Output of ith round(Ri) = 01100001110110000001110101111100
```