Name : Prem Vinod Bansod
Roll No : 41310
Assignment No : 01 (ICS)

Code:

```
#include<bits/stdc++.h>
using namespace std;
class Sdes
{
	public:
	map<int,int> key,key1,key2,p10,p8,p4,pt,expanded,ct;
	int *Larr,*Rarr,*IPLarr,*IPRarr,*S0L,*S1R;
	int s0[4][4] = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
	int s1[4][4] = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
	Sdes()
	{
		Larr = new int[5];
		Rarr = new int[5];
		IPLarr = new int[4];
		IPRarr = new int[4];
		S0L = new int[4];
		S1R = new int[4];
	}
	void inputKey()
	{
		cout<<"Enter 10 bit key"<<endl;
		int x;
		for(int i = 1;i<=10;i++)
		{
			cin>>x;
			key[i] = x;
		}
	}
	void print(map<int,int> key)
	{
		for(auto i:key)
		{
			cout<<i.second;
		}
		cout<<endl;
	}
	void permute10()
	{
		int order[10] = {3,5,2,7,4,10,1,9,8,6};
		for(int i = 0;i<10;i++)
		{
			p10[i+1] = key[order[i]];
		}
	}
	void inputLArray(map<int,int> temp)
	{
```

```
	for(int i = 0;i<5;i++)
	{
		Larr[i] = temp[i+1];
	}
}
void inputRArray(map<int,int> temp)
{
	for(int i = 0;i<5;i++)
	{
		Rarr[i] = temp[i+1+5];
	}
}
void printArray(int arr[],int n)
{
	for(int i = 0;i<n;i++)
	{
		cout<<arr[i];
	}
	cout<<endl;
}
map<int,int> permute8()
{
	int order[8] = {6,3,7,4,8,5,10,9};
	for(int i = 0;i<8;i++)
	{
		p8[i+1] = key[order[i]];
	}
	return p8;
}
void assignArrayToKey()
{
	int j = 0;
	for(int i = 1;i<=10;i++)
	{
		if(j<5)
		{
			key[i] = Larr[j];
		}
		else
		{
			key[i] = Rarr[j-5];
		}
		j++;
	}
}
void inputPlainText()
{
	cout<<"Enter 8 bit Plain Text"<<endl;
	int x;
	for(int i = 1;i<=8;i++)
	{
		cin>>x;
```

```
		pt[i] = x;
	}

}
map<int,int> initialPermutation(map<int,int> inp)
{
	map<int,int> temp;
	int order[8] = {2,6,3,1,4,8,5,7};
	for(int i = 0;i<8;i++)
	{
		temp[i+1] = inp[order[i]];
	}
	return temp;
}
void InputInitialPermutationLeftArray(map<int,int> inp)
{
	for(int i = 0;i<4;i++)
	{
		IPLarr[i] = inp[i+1];
	}
}
void InputInitialPermutationRightArray(map<int,int> inp)
{
	for(int i = 0;i<4;i++)
	{
		IPRarr[i] = inp[i+1+4];
	}
}
void InputS0L(map<int,int> temp)
{
	for(int i = 0;i<4;i++)
	{
		S0L[i] = temp[i+1];
	}
}
void InputS1R(map<int,int> temp)
{
	for(int i = 0;i<4;i++)
	{
		S1R[i] = temp[i+1+4];
	}
}
void expandpermuted()
{
	int order[8] = {4,1,2,3,2,3,4,1};
	for(int i = 0;i<8;i++)
	{
		expanded[i+1] = IPRarr[order[i]-1];
	}
}
void permute4(string s)
{
```

```
		int order[4] = {2,4,3,1};
		map<int,int> temp;
		for(int i = 0;i<4;i++)
		{
			temp[i+1] = int(s[i])-48;
		}
		for(int i = 0;i<4;i++)
		{
			p4[i+1] = temp[order[i]];
		}
	}
	void ipinverse()
	{
		map<int,int> temp;
		int j = 0;
		int order[8] = {4,1,3,5,7,2,8,6};
		for(int i = 0;i<8;i++)
		{
			if(j<4)
			{
				temp[i+1] = Larr[j];
			}
			else
			{
				temp[i+1] = Rarr[j-4];
			}
			j++;
		}
		for(int i = 0;i<8;i++)
		{
			ct[i+1] = temp[order[i]];
		}
	}
};
int* leftRotatebyOne(int arr[], int n)
{
	int temp = arr[0], i;
	for (i = 0; i < n - 1; i++)
		arr[i] = arr[i + 1];

	arr[n-1] = temp;
	return arr;
}
int* leftRotate(int arr[], int d, int n)
{
	for (int i = 0; i < d; i++)
		arr = leftRotatebyOne(arr, n);
	return arr;
}
map<int,int> xor8(map<int,int> temp1,map<int,int> temp2)
{
		map<int,int> ans;
```

```
		for(int i = 1;i<=8;i++)
		{
			ans[i] = temp1[i]^temp2[i];
		}
		return ans;
}
int binaryToDecimal(string n)
{
	string num = n;
	int dec_value = 0;
	int base = 1;
	int len = num.length();
	for (int i = len - 1; i >= 0; i--) {
		if (num[i] == '1')
		    dec_value += base;
		base = base * 2;
	}

	return dec_value;
}
string intToString(int x,int y)
{
		stringstream ss;
		ss<<x<<y;
		string s;
		ss>>s;
		return s;
}
string decToBinary(int n)
{
		if(n == 0)
		{
			return "00";
		}
		if(n == 1)
		{
			return "01";
		}
		if(n == 2)
		{
			return "10";
		}
		if(n == 3)
		{
			return "11";
		}
		return "";
}
string calculate(int *arr,int temp[4][4])
{
	string s = intToString(arr[0],arr[3]);
	int row = binaryToDecimal(s);
```

```
	s = intToString(arr[1],arr[2]);
	int col = binaryToDecimal(s);
	string ans = decToBinary(temp[row][col]);
	return ans;
}
int* xor4(map<int,int> temp1,int *temp2)
{
		int *ans = new int[4];
		for(int i = 0;i<4;i++)
		{
			ans[i] = temp1[i+1]^temp2[i];
		}
		return ans;
}
int main()
{
	map<int,int> temp;
	Sdes obj;
	obj.inputKey();
	cout<<"Key = ";
	obj.print(obj.key);
	obj.permute10();
	cout<<"After Permute 10 = ";
	obj.print(obj.p10);
	obj.inputLArray(obj.p10);
	obj.inputRArray(obj.p10);
	cout<<"Left Array = ";
	obj.printArray(obj.Larr,5);
	cout<<"Right Array = ";
	obj.printArray(obj.Rarr,5);
	cout<<"After Rotation\n";
	obj.Larr = leftRotate(obj.Larr,1,5);
	obj.Rarr = leftRotate(obj.Rarr,1,5);
	cout<<"Left Array = ";
	obj.printArray(obj.Larr,5);
	cout<<"Right Array = ";
	obj.printArray(obj.Rarr,5);
	cout<<"After Assigning Array to Key\n";
	obj.assignArrayToKey();
	cout<<"Key = ";
	obj.print(obj.key);
	obj.key1 = obj.permute8();
	cout<<"Key1 = ";
	obj.print(obj.key1);
	cout<<"Left Array = ";
	obj.printArray(obj.Larr,5);
	cout<<"Right Array = ";
	obj.printArray(obj.Rarr,5);
	cout<<"After Rotation\n";
	obj.Larr = leftRotate(obj.Larr,2,5);
	obj.Rarr = leftRotate(obj.Rarr,2,5);
	cout<<"Left Array = ";
```

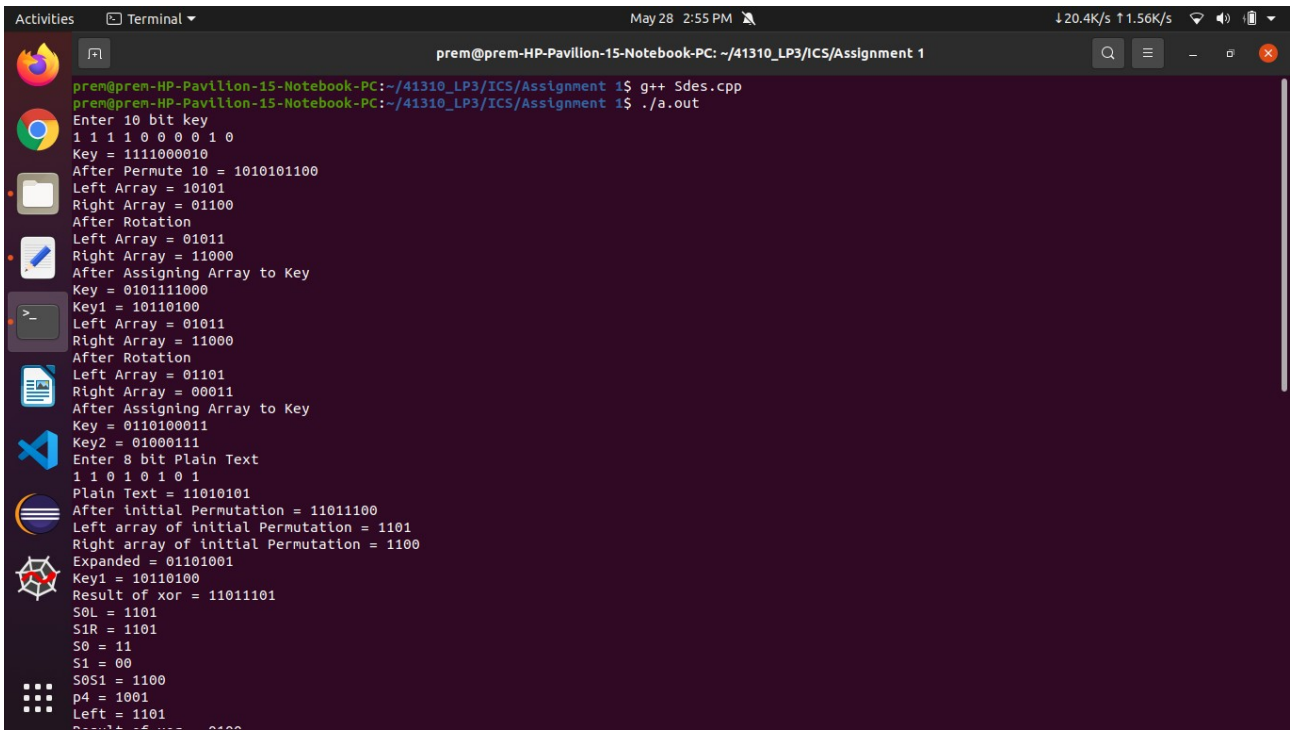
```
obj.printArray(obj.Larr,5);
cout<<"Right Array = ";
obj.printArray(obj.Rarr,5);
cout<<"After Assigning Array to Key\n";
obj.assignArrayToKey();
cout<<"Key = ";
obj.print(obj.key);
obj.key2 = obj.permute8();
cout<<"Key2 = ";
obj.print(obj.key2);
obj.inputPlainText();
cout<<"Plain Text = ";
obj.print(obj.pt);
obj.pt = obj.initialPermutation(obj.pt);
cout<<"After initial Permutation = ";
obj.print(obj.pt);
cout<<"Left array of initial Permutation = ";
obj.InputInitialPermutationLeftArray(obj.pt);
obj.printArray(obj.IPLarr,4);
cout<<"Right array of initial Permutation = ";
obj.InputInitialPermutationRightArray(obj.pt);
obj.printArray(obj.IPRarr,4);
obj.expandpermuted();
cout<<"Expanded = ";
obj.print(obj.expanded);
cout<<"Key1 = ";
obj.print(obj.key1);
temp = xor8(obj.expanded,obj.key1);
cout<<"Result of xor = ";
obj.print(temp);
obj.InputS0L(temp);
obj.InputS1R(temp);
cout<<"S0L = ";
obj.printArray(obj.S0L,4);
cout<<"S1R = ";
obj.printArray(obj.S1R,4);

string ss0 = calculate(obj.S0L,obj.s0);
cout<<"S0 = "<<ss0<<endl;

string ss1 = calculate(obj.S1R,obj.s1);
cout<<"S1 = "<<ss1<<endl;

string s0s1 = "";
s0s1.append(ss0);
s0s1.append(ss1);
cout<<"S0S1 = "<<s0s1<<endl;
obj.permute4(s0s1);
cout<<"p4 = ";
obj.print(obj.p4);

obj.Larr = obj.IPRarr;
```

```
cout<<"Left = ";
obj.printArray(obj.IPLarr,4);
obj.Rarr = xor4(obj.p4,obj.IPLarr);
cout<<"Result of xor = ";
obj.printArray(obj.Rarr,4);


obj.IPRarr = obj.Rarr;
cout<<"After Expanded = ";
obj.expandpermuted();
obj.print(obj.expanded);
cout<<"Key2 = ";
obj.print(obj.key2);
cout<<"Result of xor = ";
temp = xor8(obj.expanded,obj.key2);
obj.print(temp);

obj.InputS0L(temp);
obj.InputS1R(temp);
cout<<"S0L = ";
obj.printArray(obj.S0L,4);
cout<<"S1R = ";
obj.printArray(obj.S1R,4);

ss0 = calculate(obj.S0L,obj.s0);
cout<<"S0 = "<<ss0<<endl;

ss1 = calculate(obj.S1R,obj.s1);
cout<<"S1 = "<<ss1<<endl;

s0s1 = "";
s0s1.append(ss0);
s0s1.append(ss1);
cout<<"S0S1 = "<<s0s1<<endl;
obj.permute4(s0s1);
cout<<"p4 = ";
obj.print(obj.p4);
cout<<"Left = ";
obj.printArray(obj.Larr,4);
obj.Larr = xor4(obj.p4,obj.Larr);
cout<<"Result of xor = ";
obj.printArray(obj.Larr,4);
obj.ipinverse();
cout<<"Cipher Text = ";
obj.print(obj.ct);

obj.ct = obj.initialPermutation(obj.ct);
cout<<"After initial Permutation = ";
obj.print(obj.ct);
cout<<"Left array of initial Permutation = ";
obj.InputInitialPermutationLeftArray(obj.ct);
obj.printArray(obj.IPLarr,4);
```

```
cout<<"Right array of initial Permutation = ";
obj.InputInitialPermutationRightArray(obj.ct);
obj.printArray(obj.IPRarr,4);
obj.expandpermuted();
cout<<"Expanded = ";
obj.print(obj.expanded);
cout<<"Key2 = ";
obj.print(obj.key2);
temp = xor8(obj.expanded,obj.key2);
cout<<"Result of xor = ";
obj.print(temp);
obj.InputS0L(temp);
obj.InputS1R(temp);
cout<<"S0L = ";
obj.printArray(obj.S0L,4);
cout<<"S1R = ";
obj.printArray(obj.S1R,4);

ss0 = calculate(obj.S0L,obj.s0);
cout<<"S0 = "<<ss0<<endl;

ss1 = calculate(obj.S1R,obj.s1);
cout<<"S1 = "<<ss1<<endl;

s0s1 = "";
s0s1.append(ss0);
s0s1.append(ss1);
cout<<"S0S1 = "<<s0s1<<endl;
obj.permute4(s0s1);
cout<<"p4 = ";
obj.print(obj.p4);

cout<<"Left = ";
obj.printArray(obj.Larr,4);
obj.Rarr = xor4(obj.p4,obj.IPLarr);
cout<<"Result of xor = ";
obj.printArray(obj.Rarr,4);
obj.Larr = obj.IPRarr;

obj.IPRarr = obj.Rarr;
cout<<"After Expanded = ";
obj.expandpermuted();
obj.print(obj.expanded);
cout<<"Key1 = ";
obj.print(obj.key1);
cout<<"Result of xor = ";
temp = xor8(obj.expanded,obj.key1);
obj.print(temp);

obj.InputS0L(temp);
obj.InputS1R(temp);
cout<<"S0L = ";
```

```
        obj.printArray(obj.S0L,4);
        cout<<"S1R = ";
        obj.printArray(obj.S1R,4);

        ss0 = calculate(obj.S0L,obj.s0);
        cout<<"S0 = "<<ss0<<endl;

        ss1 = calculate(obj.S1R,obj.s1);
        cout<<"S1 = "<<ss1<<endl;

        s0s1 = "";
        s0s1.append(ss0);
        s0s1.append(ss1);
        cout<<"S0S1 = "<<s0s1<<endl;
        obj.permute4(s0s1);
        cout<<"p4 = ";
        obj.print(obj.p4);
        cout<<"Left = ";
        obj.printArray(obj.Larr,4);
        obj.Larr = xor4(obj.p4,obj.Larr);
        cout<<"Result of xor = ";
        obj.printArray(obj.Larr,4);
        obj.ipinverse();
        cout<<"Plain Text = ";
        obj.print(obj.ct);

        return 0;
}
```

output:



```
p4 = 1001
Left = 1101
Result of xor = 0100
After Expanded = 00101000
Key2 = 01000111
Result of xor = 01101111
S0L = 0110
S1R = 1111
S0 = 10
S1 = 11
S0S1 = 1011
p4 = 0111
Left = 1100
Result of xor = 1011
Cipher Text = 11100001
After initial Permutation = 10110100
Left array of initial Permutation = 1011
Right array of initial Permutation = 0100
Expanded = 00101000
Key2 = 01000111
Result of xor = 01101111
S0L = 0110
S1R = 1111
S0 = 10
S1 = 11
S0S1 = 1011
p4 = 0111
Left = 1011
Result of xor = 1100
After Expanded = 01101001
Key1 = 10110100
Result of xor = 11011101
S0L = 1101
S1R = 1101
S0 = 11
S1 = 00
S0S1 = 1100
p4 = 1001
```

```
Key2 = 01000111
Result of xor = 01101111
S0L = 0110
S1R = 1111
S0 = 10
S1 = 11
S0S1 = 1011
p4 = 0111
Left = 1100
Result of xor = 1011
Cipher Text = 11100001
After initial Permutation = 10110100
Left array of initial Permutation = 1011
Right array of initial Permutation = 0100
Expanded = 00101000
Key2 = 01000111
Result of xor = 01101111
S0L = 0110
S1R = 1111
S0 = 10
S1 = 11
S0S1 = 1011
p4 = 0111
Left = 1011
Result of xor = 1100
After Expanded = 01101001
Key1 = 10110100
Result of xor = 11011101
S0L = 1101
S1R = 1101
S0 = 11
S1 = 00
S0S1 = 1100
p4 = 1001
Left = 0100
Result of xor = 1101
Plain Text = 11010101
prem@prem-HP-Pavilion-15-Notebook-PC:~/41310_LP3/ICS/Assignment 1$
```