**PRACTICAL: 6**

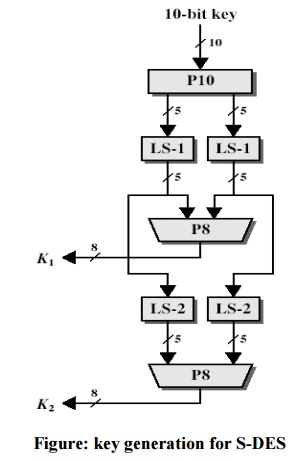
**AIM:**

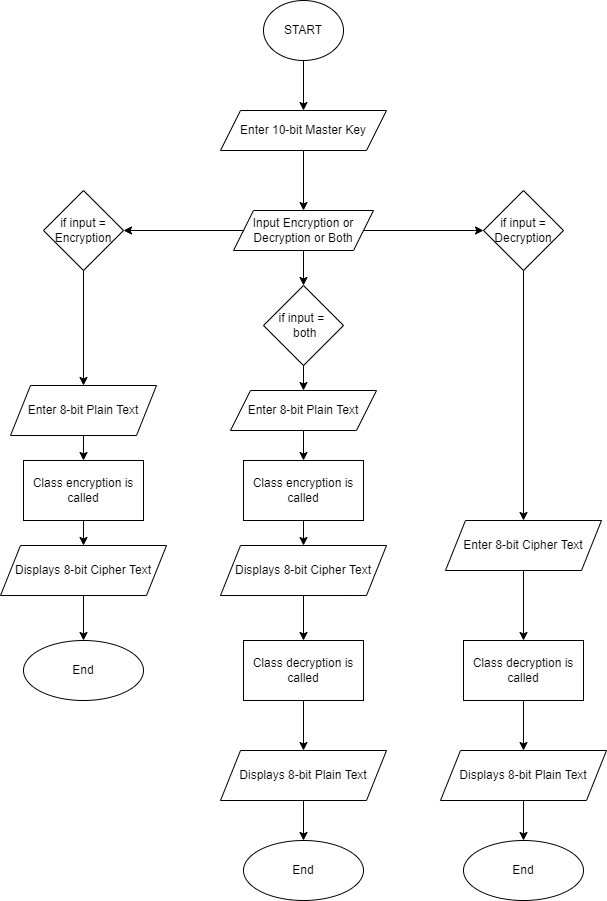
Simplified DES (S-DES) is a symmetric-key block cipher. The S-DES encryption algorithm takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8- bit block of cipher text as output. It follows two rounds. Implement S-DES symmetric encryption Algorithm.

**THEORY:** Simplified Data Encryption Standard (S-DES) is a simple version of the DES Algorithm. It is similar to the DES algorithm but is a smaller algorithm and has fewer parameters than DES. It was made for educational purposes so that understanding DES would become simpler. It is a block cipher that takes a block of plain text and converts it into ciphertext. It takes a block of 8 bit.

It is a symmetric key cipher i.e. they use the same key for both encryption and decryption. In this article, we are going to demonstrate key generation for s-des encryption and decryption algorithm. We take a random 10-bit key and produce two 8-bit keys which will be used for encryption and decryption.

Key Generation Concept: In the key generation algorithm, we accept the 10-bit key and convert it into two 8 bit keys. This key is shared between both sender and receiver.

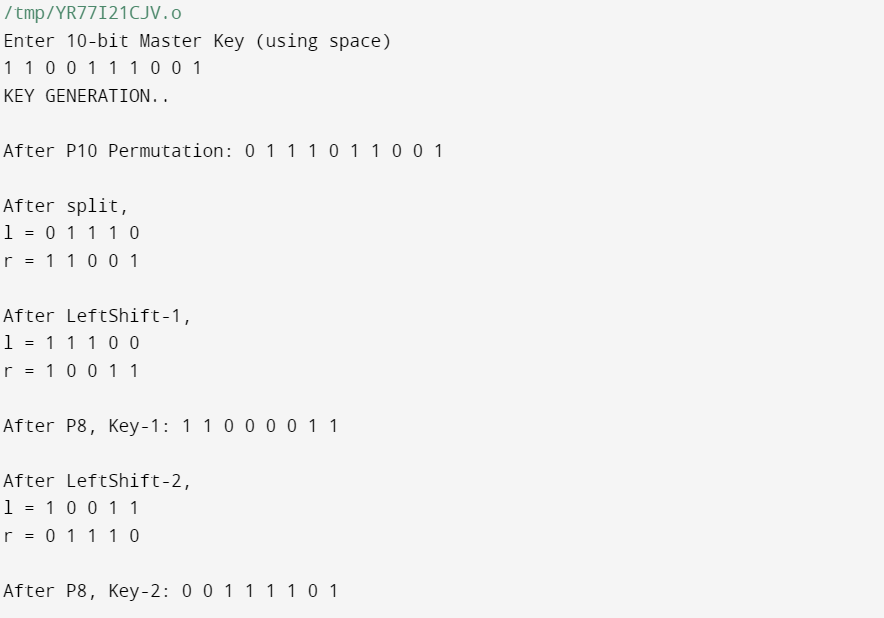




**CODE:**

|  |
| --- |
| #include <iostream>  using namespace std;  //Functions  void printArray(int arr[],int n)  {  for (int i = 0; i < n; i++)  cout << arr[i] << " ";  cout << endl;  }  void Permutation(int arr[], int index[], int n){  int temp[n];  for (int i=0; i<n; i++)  temp[i] = arr[index[i]-1];  for (int i=0; i<n; i++)  arr[i] = temp[i];  }  void ExPermutation(int arr[], int index[], int arr2[], int n){  for (int i=0; i<n; i++)  arr2[i] = arr[index[i]-1];  }    void Split(int arr[], int n, int \*l, int \*r){  for(int i=0;i<n/2;i++)  l[i] = arr[i];  for(int j=0,i=n/2;i<n;i++,j++)  r[j] = arr[i];  }    int bin2dec(int arr[],int size){  int decimal = 0 ;  for(int i = 0 ; i < size ; i++)  decimal = (decimal << 1) + arr[i] ;  return decimal;  }    void dec2bin(int opSn, int \*ar){  int i=0;  while(opSn!=0)  {  ar[i] = opSn%2;  i++;  opSn = opSn/2;  }  }    void combine(int arr1[], int arr2[], int \*arr3, int n){  for (int i=0;i<n/2;i++)  arr3[i]=arr1[i];  for (int i=n/2,j=0;i<n;i++,j++)  arr3[i]=arr2[j];  }    void S\_box(int a[],int b[],int \*opS0S1){  int S0[4][4] = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}}, S1[4][4] = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};  //S0  int rowS0bin[2] = {a[0],a[3]}, colS0bin[2] = {a[1],a[2]};  int rowS0dec = bin2dec(rowS0bin,2), colS0dec = bin2dec(colS0bin,2);  int opS0dec = S0[rowS0dec][colS0dec];  int opS0bin[2]={};  dec2bin(opS0dec, opS0bin);    //S1  int rowS1bin[2] = {b[0],b[3]}, colS1bin[2] = {b[1],b[2]};  int rowS1dec = bin2dec(rowS1bin,2), colS1dec = bin2dec(colS1bin,2);  int opS1dec = S1[rowS1dec][colS1dec];  int opS1bin[2]={};  dec2bin(opS1dec, opS1bin);    for (int i=0;i<2;i++)  opS0S1[i]=opS0bin[i];  for (int i=2,j=0;i<4;i++,j++)  opS0S1[i]=opS1bin[j];  cout<<"After S-Box: ";  printArray(opS0S1,4);  cout<<endl;  }    void Swap(int \*left\_array, int \*right\_array, int n){  int temp[n];  for (int i=0; i<n; i++)  temp[i] = left\_array[i];  for (int i=0; i<n; i++)  left\_array[i]= right\_array[i];  for (int i=0; i<n; i++)  right\_array[i]= temp[i];      }    void XOR(int arr1[],int arr2[],int n){  int temp[n];  for(int i=0; i<n; i++)  {  temp[i] = arr1[i] ^ arr2[i];  }  for (int i=0; i<n; i++)  arr2[i] = temp[i];  }    void leftRotate(int arr[], int d, int n)  {  int temp[d];  for (int i=0; i<d; i++)  temp[i] = arr[i];  for (int i = 0; i < n-d; i++)  arr[i] = arr[i+d];  for (int i=n-d,j=0; i<n; i++,j++)  arr[i]=temp[j];    }  class KeyGeneration {  private:  int P10\_rule[10] = {3,5,2,7,4,10,1,9,8,6};  int P8\_rule[8] = {6,3,7,4,8,5,10,9};  int temp\_left[5]={}, temp\_right[5]={};    public:  KeyGeneration(){  cout<<endl;  cout<<"KEY GENERATION.."<<endl;  cout<<endl;  }  void key(int master\_key[], int \*k1, int \*k2){  //P10  Permutation(master\_key,P10\_rule,10);  cout<<"After P10 Permutation: ";  printArray(master\_key,10);  cout<<endl;  //Split  Split(master\_key,10,temp\_left,temp\_right);  cout<<"After split, "<<endl;  cout<<"l = ";  printArray(temp\_left,5);  cout<<"r = " ;  printArray(temp\_right,5);  cout<<endl;  //LS-1  leftRotate(temp\_left,1,5);  leftRotate(temp\_right,1,5);  cout<<"After LeftShift-1, "<<endl;  cout<<"l = ";  printArray(temp\_left,5);  cout<<"r = ";  printArray(temp\_right,5);  cout<<endl;  //P-8  combine(temp\_left,temp\_right,master\_key,10);  Permutation(master\_key,P8\_rule,10);  cout<<"After P8, Key-1: ";  for(int i=0;i<8;i++)  k1[i]=master\_key[i];  printArray(k1,8);  cout<<endl;  //LS-2  leftRotate(temp\_left,2,5);  leftRotate(temp\_right,2,5);  cout<<"After LeftShift-2, "<<endl;  cout<<"l = ";  printArray(temp\_left,5);  cout<<"r = ";  printArray(temp\_right,5);  cout<<endl;  //P-8  combine(temp\_left,temp\_right,master\_key,10);  Permutation(master\_key,P8\_rule,10);  cout<<"After P8, Key-2: ";  for(int i=0;i<8;i++)  k2[i]=master\_key[i];  printArray(k2,8);  }  };  class Roundfunction{  private:  int Expanrule[8] = {4,1,2,3,2,3,4,1};  int P4\_rule[4] = {2,4,3,1};  int r\_arr2[8]={},a[4]={},b[4]={};  int opS0S1[4]={};    public:  void roundfun(int \*k1,int \*l\_arr, int \*r\_arr, int \*fk1){  ExPermutation(r\_arr, Expanrule, r\_arr2,8);  cout<<"After EP: ";  printArray(r\_arr2,8);  cout<<endl;    //XOR with K1  XOR(k1,r\_arr2,8);  cout<<"XOR with key"<<endl;  printArray(r\_arr2,8);  cout<<endl;  //Split  Split(r\_arr2,8,a,b);  cout<<"After Split"<<endl;  cout<<"l = ";  printArray(a,4);  cout<<"r = ";  printArray(b,4);  cout<<endl;  //Sbox  S\_box(a,b,opS0S1);  //P4  Permutation(opS0S1,P4\_rule,4);  cout<<"After P4"<<endl;  printArray(opS0S1,4);  cout<<endl;  //XOR with left array  XOR(opS0S1,l\_arr,4);  cout<<"XOR with leftarray"<<endl;  printArray(l\_arr,4);  cout<<endl;  //combine  combine(l\_arr,r\_arr,fk1,8);  cout<<"After combine"<<endl;  printArray(fk1,8);  cout<<endl;    }  };  class encrypt : public Roundfunction{  private:  int IP\_rule[8] = {2,6,3,1,4,8,5,7};  int IP\_inv\_rule[8] = {4,1,3,5,7,2,8,6};  int fkk[8]={};    public:  encrypt(){  cout<<endl;  cout<<"ENCRYPTING.."<<endl;  cout<<endl;  }  int l\_arr[4]={},r\_arr[4]={};  //IP  void enc(int arr[], int \*key1, int \*key2, int \*fk1){    Permutation(arr, IP\_rule, 8);  cout<<"After IP: ";  printArray(arr,8);  cout<<endl;  //Split  Split(arr,8, l\_arr,r\_arr);  cout<<"After split, "<<endl;  cout<<"l = ";  printArray(l\_arr,4);  cout<<"r = " ;  printArray(r\_arr,4);  cout<<endl;  //fk1  cout<<"Round Function(fk)-1"<<endl;  Roundfunction::roundfun(key1,l\_arr,r\_arr,fk1);  //Swap  Swap(l\_arr,r\_arr,4);  cout<<"After Swap"<<endl;  cout<<"l = ";  printArray(l\_arr,4);  cout<<"r = " ;  printArray(r\_arr,4);  cout<<endl;  //fk2  cout<<"Round Function(fk)-2"<<endl;  Roundfunction::roundfun(key2,l\_arr,r\_arr,fk1);  //ipinv  Permutation(fk1,IP\_inv\_rule,8);  cout<<"After IP-Inverse, 8-bit Cipher Text is: "<<endl;  printArray(fk1,8);  }  };  class decrypt : public Roundfunction{  private:  int IP\_rule[8] = {2,6,3,1,4,8,5,7};  int IP\_inv\_rule[8] = {4,1,3,5,7,2,8,6};  int fkk[8]={};    public:  int l\_arr[4]={},r\_arr[4]={};    //IP  void decryp(int arr[], int \*key1, int \*key2, int \*fk1){    Permutation(arr, IP\_rule, 8);  cout<<"IP"<<endl;  printArray(arr,8);  //Split  Split(arr,8, l\_arr,r\_arr);  cout<<"Split"<<endl;  printArray(l\_arr,4);  printArray(r\_arr,4);  //fk1  Roundfunction::roundfun(key2,l\_arr,r\_arr,fk1);  //Swap  Swap(l\_arr,r\_arr,4);  cout<<"swap"<<endl;  printArray(l\_arr,4);  printArray(r\_arr,4);  //fk2  Roundfunction::roundfun(key1,l\_arr,r\_arr,fk1);  //ipinv  Permutation(fk1,IP\_inv\_rule,8);  cout<<"After IP-Inverse, 8-bit Plain Text is: "<<endl;  printArray(fk1,8);  }  };  int main()  {  char input;  int arr[8]={};  int master\_key[10]={};  int k1[8]={},k2[8]={};  int fk1[8] = {};    //Key  cout<<"Enter 10-bit Master Key (using space)"<<endl;  for(int i=0;i<10;i++){  cin>>master\_key[i];  }  if((sizeof(master\_key)/sizeof(master\_key[0]))!=10)  throw "Error. Enter 10-bits";  KeyGeneration k;  k.key(master\_key,k1,k2);  cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;  cout<<endl;  cout<<"Enter e for Encryption | Enter d for Decryption | Enter b for Both"<<endl;  cin>>input;  if (input == 'b'||input == 'B'){  cout<<"Enter 8-bit Plain Text (using space)"<<endl;  for(int i=0;i<8;i++){  cin>>arr[i];  }  if((sizeof(arr)/sizeof(arr[0]))!=8)  throw "Error. Enter 8-bits";  encrypt e;  e.enc(arr,k1,k2,fk1);  for(int i=0;i<8;i++)  arr[i] = fk1[i];  cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;  decrypt d;  d.decryp(arr,k1,k2,fk1);  }  else if (input == 'e'||input == 'E'){  cout<<"Enter 8-bit Plain Text (using space)"<<endl;  for(int i=0;i<8;i++){  cin>>arr[i];  }  if((sizeof(arr)/sizeof(arr[0]))!=8)  throw "Error. Enter 8-bits";  encrypt e;  e.enc(arr,k1,k2,fk1);  }  else if (input == 'd'||input == 'D'){  cout<<"Enter 8-bit Cipher Text (using space)"<<endl;  for(int i=0;i<8;i++){  cin>>arr[i];  }  if((sizeof(arr)/sizeof(arr[0]))!=8)  throw "Error. Enter 8-bits";  decrypt d;  d.decryp(arr,k1,k2,fk1);  }  else  throw "Error, Choose correct option";    return 0;  } |

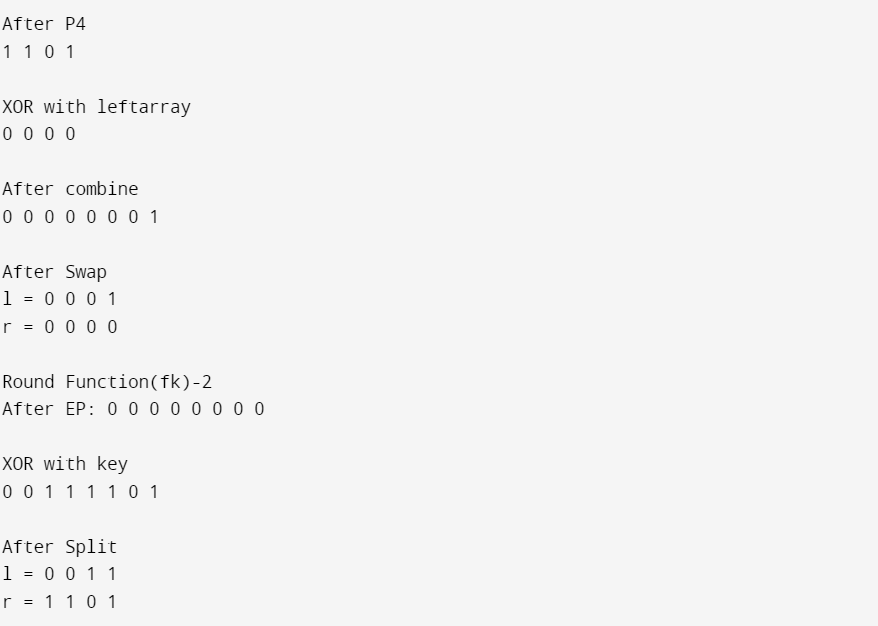
**OUTPUT:**

****

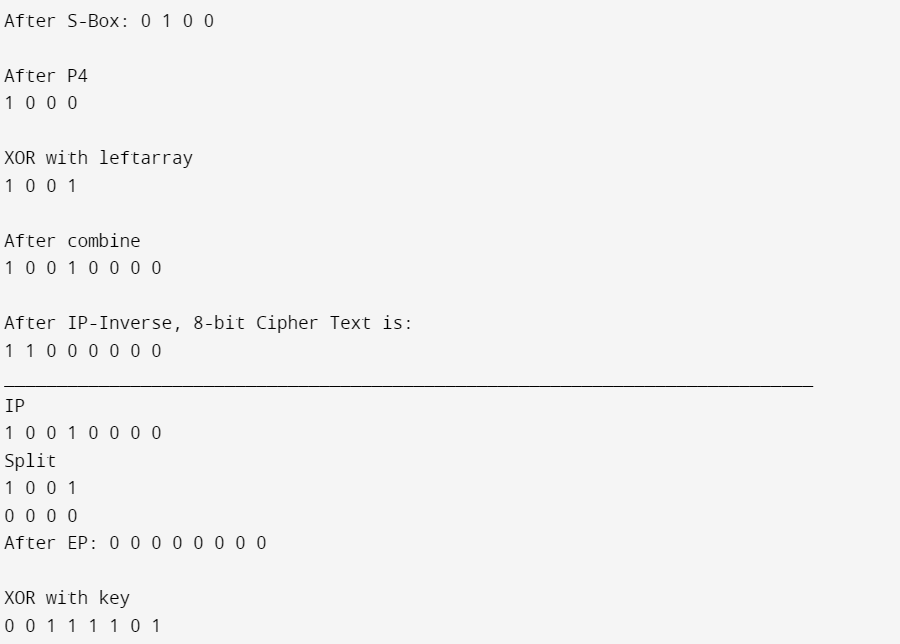
First we have to enter 10 digit master key which is random

****

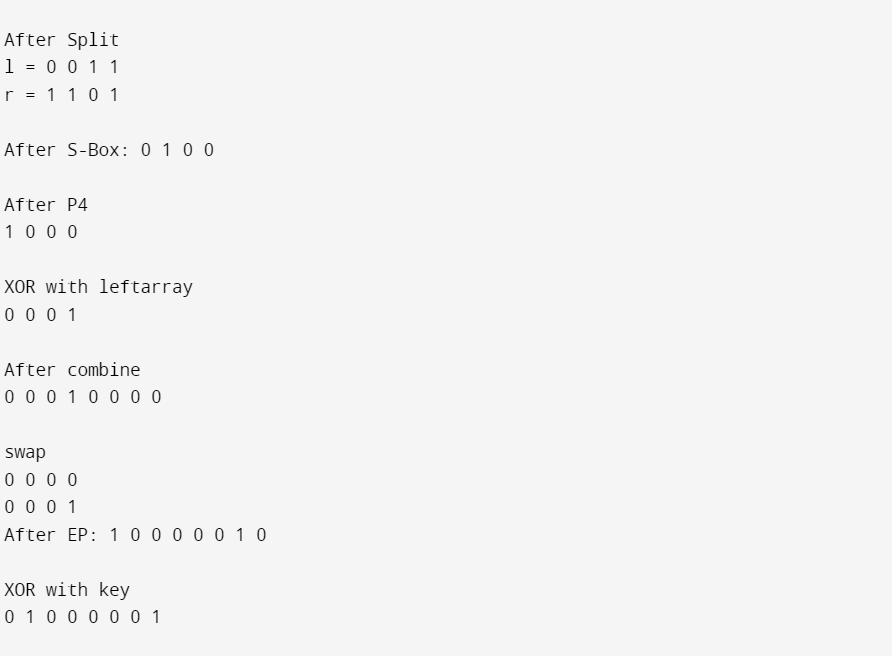
It will ask for encryption decryption and for both

****

It will perform several steps for encryption and descryption

****

It will generate 8 bit cipher text

****

****

Finally give output of 8 bit plain Text

**LATEST APPLICATIONS:** Use of the DES algorithm was made mandatory for all financial transactions of the U.S. government involving electronic fund transfer, including those conducted by member banks of the Federal Reserve System.

**LEARNING OUTCOME:**

From this practical we get to know about the Very powerful algorithm in which it is nearly impossible to crack for anyone and how the complexity is there to construct or design.

**REFERENCES:**

1. Geeks fot geeks: <https://geeksforgeeks.org>