**PRACTICAL - 03**

**Code:**

// Ashwin Navange A-38 CSE

#include<bits/stdc++.h>

using namespace std;

static int round1\_key[8],round2\_key[8];

void p10(int key[])

{

int out[10]={3,5,2,7,4,10,1,9,8,6};

int temp[10];

for(int i=0;i<10;i++)

temp[i]=key[i];

for(int i=0;i<10;i++)

{

key[i]=temp[out[i]-1];

}

}

void p8(int key[])

{

int out[8]={6,3,7,4,8,5,10,9};

int temp[10];

for(int i=0;i<10;i++)

temp[i]=key[i];

for(int i=0;i<8;i++)

{

key[i]=temp[out[i]-1];

}

}

void p4(int s0s1[])

{

int out[4]={2,4,3,1};

int temp[4];

for(int i=0;i<4;i++)

temp[i]=s0s1[i];

for(int i=0;i<4;i++)

{

s0s1[i]=temp[out[i]-1];

}

}

void left\_shift(int left\_half[], int right\_half[], int shift\_count)

{

int temp1=left\_half[0];

int temp2=right\_half[0];

for(int i=0;i<4;i++)

{

left\_half[i]=left\_half[i+1];

right\_half[i]=right\_half[i+1];

}

left\_half[4]=temp1;

right\_half[4]=temp2;

if(shift\_count==2)

left\_shift(left\_half,right\_half,1);

}

int\* generate\_key(int key[],int round)

{

int left\_half[5],right\_half[5];

static int key1[10],key2[8];

p10(key);

for(int i=0;i<10;i++)

{

if(i<5)

{

left\_half[i]=key[i];

}

else

{

right\_half[i-5]=key[i];

}

}

left\_shift(left\_half,right\_half,1);

for(int i=0;i<5;i++)

{

key1[i]=left\_half[i];

key1[i+5]=right\_half[i];

}

if(round==1)

{

p8(key1);

return key1;

}

else

{

left\_shift(left\_half,right\_half,2);

for(int i=0;i<5;i++)

{

key2[i]=left\_half[i];

key2[i+5]=right\_half[i];

}

p8(key2);

return key2;

}

}

void initial\_permutation(int pt[])

{

int out[8]={2,6,3,1,4,8,5,7};

int temp[8];

for(int i=0;i<8;i++)

temp[i]=pt[i];

for(int i=0;i<8;i++)

{

pt[i]=temp[out[i]-1];

}

}

void inverse\_initial\_permutation(int pt[])

{

int out[8]={2,6,3,1,4,8,5,7};

int temp[8];

for(int i=0;i<8;i++)

temp[i]=pt[i];

for(int i=0;i<8;i++)

{

pt[out[i]-1]=temp[i];

}

}

int\* expand\_and\_permute(int right\_half[])

{

int out[8]={4,1,2,3,2,3,4,1};

int temp[4];

static int expanded\_right[8];

for(int i=0;i<4;i++)

temp[i]=right\_half[i];

for(int i=0;i<8;i++)

{

expanded\_right[i]=temp[out[i]-1];

}

return expanded\_right;

}

int get\_S0(int row,int column)

{

int s0[4][4]={

{01,00,11,10},

{11,10,01,00},

{00,10,01,11},

{11,01,11,10}

};

return s0[row][column];

}

int get\_S1(int row,int column)

{

int s1[4][4]={

{00,01,10,11},

{10,00,01,11},

{11,00,01,00},

{10,01,00,11}

};

return s1[row][column];

}

int\* rounds(int pt[],int key[],int round\_no,int flag)

{

int left[4],right[4],\*expanded\_right,s0[4],s1[4],temp\_key[10];

cout<<"\nROUND-"<<round\_no;

for(int i=0;i<10;i++)

{

temp\_key[i]=key[i];

}

if(round\_no==1)

initial\_permutation(pt);

for(int i=0;i<4;i++)

{

left[i]=pt[i];

right[i]=pt[i+4];

}

expanded\_right= expand\_and\_permute(right);

static int\* key1;

if(flag==0)

{ key1=generate\_key(key,round\_no);

if(round\_no==1)

{

for(int i=0;i<8;i++)

round1\_key[i]=key1[i];

}

else

{

for(int i=0;i<8;i++)

round2\_key[i]=key1[i];

}

cout<<"\n\nEncode Key of Round "<<round\_no<<": ";

for(int i=0;i<8;i++)

{

cout<<key1[i];

}

}

else

{

if(round\_no==1)

{

for(int i=0;i<8;i++)

{

key1[i]=round2\_key[i];

}

}

else

{

for(int i=0;i<8;i++)

{

key1[i]=round1\_key[i];

}

}

cout<<"\n\nDecode Key of Round "<<round\_no<<": ";

for(int i=0;i<8;i++)

{

cout<<key1[i];

}

}

for(int i=0;i<8;i++)

{

expanded\_right[i]=expanded\_right[i] ^ key1[i];

if(i<4)

s0[i]=expanded\_right[i];

else

s1[i-4]=expanded\_right[i];

}

int row=s0[3]+(s0[0]\*2);

int column=s0[2]+(s0[1]\*2);

static int s0s1[4];

int ss0=get\_S0(row,column);

row=s1[3]+(s1[0]\*2);

column=s1[2]+(s1[1]\*2);

int ss1=get\_S1(row,column);

s0s1[1]=ss0%10;

s0s1[0]=ss0/10;

s0s1[3]=ss1%10;

s0s1[2]=ss1/10;

p4(s0s1);

static int new\_plain\_text[8];

for(int i=0;i<4;i++)

{

s0s1[i]=s0s1[i] ^ left[i];

if(round\_no!=2)

{

new\_plain\_text[i]=right[i];

new\_plain\_text[i+4]=s0s1[i];

}

else

{

new\_plain\_text[i+4]=right[i];

new\_plain\_text[i]=s0s1[i];

}

}

cout<<"\nRound "<<round\_no<<" Output: ";

for(int i=0;i<8;i++)

cout<<new\_plain\_text[i];

cout<<endl;

if(round\_no==1)

{

if(flag==0)

rounds(new\_plain\_text,temp\_key,2,0);

else

rounds(new\_plain\_text,temp\_key,2,1);

}

else

{

return new\_plain\_text;

}

}

int\* encode(int pt[],int\* round\_text,int key[])

{

round\_text=rounds(pt,key,1,0);

inverse\_initial\_permutation(round\_text);

cout<<"\n\n----------FINAL CIPHER TEXT----------\n";

for(int i=0;i<8;i++)

cout<<round\_text[i];

return round\_text;

}

void decode(int pt[], int\* cipher\_text,int key[])

{

int \*new\_ct=rounds(cipher\_text,key,1,1);

inverse\_initial\_permutation(new\_ct);

cout<<"\n\n-------------DECODED TEXT-------------\n";

for(int i=0;i<8;i++)

cout<<new\_ct[i];

cout<<endl;

}

int main()

{

int \*round\_text, \*cipher\_text, pt[8],key[10];

string plaintext,keystring;

cout<<"Ashwin Navange A-38 CSE\n";

cout<<"\nEnter the plain text (8-bits): ";

getline(cin,plaintext);

for(int i=0;i<8;i++)

pt[i]=plaintext[i]-'0';

cout<<"Enter the key (10-bits): ";

getline(cin,keystring);

for(int i=0;i<10;i++)

key[i]=keystring[i]-'0';

cout<<"\n-------------ENCRYPTION-------------\n";

cipher\_text=encode(pt,round\_text,key);

cout<<"\n\n\n-------------DECRYPTION-------------\n";

decode(pt,cipher\_text,key);

return 0;

}

**Output:**

