NAME: SAKSHI VEER

ROLL NO: 4101066

BATCH: BA2

**Problem Statement:** **To implement a Simplified Data Encryption Standard (S-DES) algorithm.**

**CODE:**

**package** ics;

**import** java.awt.datatransfer.StringSelection;

**import** java.io.DataInputStream;

**public** **class** SDES {

**public** **int** K1,K2;

//For performing P10 permutation

**public** **static** **final** **int** ***P10***[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6};

**public** **static** **final** **int** ***P10max*** = 10;

//10 to 8 bit permutation

**public** **static** **final** **int** ***P8***[] = { 6, 3, 7, 4, 8, 5, 10, 9};

**public** **static** **final** **int** ***P8max*** = 10;

**public** **static** **final** **int** ***P4***[] = { 2, 4, 3, 1};

**public** **static** **final** **int** ***P4max*** = 4;

//Performing initial permutaion

**public** **static** **final** **int** ***IP***[] = { 2, 6, 3, 1, 4, 8, 5, 7};

**public** **static** **final** **int** ***IPmax*** = 8;

//performing final permuttaion(Inverse of initial permutation)

**public** **static** **final** **int** ***IPI***[] = { 4, 1, 3, 5, 7, 2, 8, 6};

**public** **static** **final** **int** ***IPImax*** = 8;

//expansion/permutation operation:Used in F1 Function

**public** **static** **final** **int** ***EP***[] = { 4, 1, 2, 3, 2, 3, 4, 1};

**public** **static** **final** **int** ***EPmax*** = 4;

**public** **static** **final** **int** ***S0***[][] = {{ 1, 0, 3, 2},{ 3, 2, 1, 0},{ 0, 2, 1,

3},{ 3, 1, 3, 2}};

**public** **static** **final** **int** ***S1***[][] = {{ 0, 1, 2, 3},{ 2, 0, 1, 3},{ 3, 0, 1,

2},{ 2, 1, 0, 3}};

/\*

\* Generic Function to perform all permutations

\*/

**public** **static** **int** permute(**int** x, **int** p[], **int** pmax) {

**int** y=0;

**for**(**int** i=0;i<p.length;i++) {

y=y<<1;

y=y|(x>>(pmax-p[i]))&1;

}

**return** y;

}

**public** **static** **int** F( **int** R, **int** K)

{

**int** t = *permute*( R, ***EP***, ***EPmax***) ^ K;

**int** t0 = (t >> 4) & 0xF;

**int** t1 = t & 0xF;

t0 = ***S0***[ ((t0 & 0x8) >> 2) | (t0 & 1) ][ (t0 >> 1) & 0x3 ];

t1 = ***S1***[ ((t1 & 0x8) >> 2) | (t1 & 1) ][ (t1 >> 1) & 0x3 ];

t = *permute*( (t0 << 2) | t1, ***P4***, ***P4max***);

**return** t;

}

**public** **static** **int** fK( **int** m, **int** K)

{

**int** L = (m >> 4) & 0xF;

**int** R = m & 0xF;

**return** ((L ^ *F*(R,K)) << 4) | R;

}

**public** **static** **int** SW( **int** x)

{

**return** ((x & 0xF) << 4) | ((x >> 4) & 0xF);

}

/\*

\* Ciphertext = IP-1 (fK2 (SW (fk1 (IP (plaintext))))

\*/

**public** **byte** encrypt( **int** m)

{

System.***out***.println("\nEncryption Process Starts........\n\n");

m = *permute*( m, ***IP***, ***IPmax***);

System.***out***.print("\nAfter Initial Permutation(IP) : ");

*printData*( m, 8);

m = *fK*( m, K1);

System.***out***.print("\nbefore Swap : ");

*printData*( m, 8);

m = *SW*( m);

System.***out***.print("\nAfter Swap : ");

*printData*( m, 8);

m = *fK*( m, K2);

System.***out***.print("\nbefore IP inverse : ");

*printData*( m, 8);

m = *permute*( m, ***IPI***, ***IPImax***);

**return** (**byte**) m;

}

/\*

\* Plaintext = IP-1 (fK1 (SW (fk2 (IP (ciphertext)))))

\*/

**public** **byte** decrypt( **int** m)

{

System.***out***.println("\nDecryption Process Starts........\n\n");

*printData*( m, 8);

m = *permute*( m, ***IP***, ***IPmax***);

System.***out***.print("\nAfter Permutation : ");

*printData*( m, 8);

m = *fK*( m, K2);

System.***out***.print("\nbefore Swap : ");

*printData*( m, 8);

m = *SW*( m);

System.***out***.print("\nAfter Swap : ");

*printData*( m, 8);

m = *fK*( m, K1);

System.***out***.print("\nBefore Extraction Permutation : ");

*printData*( m, 4);

m = *permute*( m, ***IPI***, ***IPImax***);

System.***out***.print("\nAfter Extraction Permutation : ");

*printData*( m, 8);

**return** (**byte**) m;

}

**public** **static** **void** printData( **int** x, **int** n)

{

**int** mask = 1 << (n-1);

**while**( mask > 0)

{

System.***out***.print( ((x & mask) == 0) ? '0' : '1');

mask >>= 1;

}

}

/\*

\*

\* K1 = P8 (Shift\_1 (P10 (Key)))

K2 = P8 (Shift\_2 (shift\_1 (P10 (Key))))

\*/

**public** SDES( **int** K) //COnstructor generates key k1 & k2 used for encyption

{

K = *permute*( K, ***P10***, ***P10max***);

**int** t1 = (K >> 5) & 0x1F;

**int** t2 = K & 0x1F;

t1 = ((t1 & 0xF) << 1) | ((t1 & 0x10) >> 4);

t2 = ((t2 & 0xF) << 1) | ((t2 & 0x10) >> 4);

K1 = *permute*( (t1 << 5)| t2, ***P8***, ***P8max***);

t1 = ((t1 & 0x7) << 2) | ((t1 & 0x18) >> 3);

t2 = ((t2 & 0x7) << 2) | ((t2 & 0x18) >> 3);

K2 = *permute*( (t1 << 5)| t2, ***P8***, ***P8max***);

}

@SuppressWarnings("deprecation")

**public** **static** **void** main( String args[]) **throws** Exception

{

DataInputStream inp=**new** DataInputStream(System.***in***);

System.***out***.println("Enter the 10 Bit Key :"); //1011011010

**int** K = Integer.*parseInt*(inp.~~readLine~~(),2);

SDES A = **new** SDES( K);

System.***out***.println("Enter the 8 Bit message To be Encrypt : ");

**int** m = Integer.*parseInt*(inp.~~readLine~~(),2); //10110110

System.***out***.print("\nKey K1: ");

SDES.*printData*( A.K1, 8);

System.***out***.print("\nKey K2: ");

SDES.*printData*( A.K2, 8);

m = A.encrypt( m);

System.***out***.print("\nEncrypted Message: ");

SDES.*printData*( m, 8);

m = A.decrypt( m);

System.***out***.print("\nDecrypted Message: ");

SDES.*printData*( m, 8);

}

}

**OUTPUT:**

Enter the 10 Bit Key :

1011011010

Enter the 8 Bit message To be Encrypt :

10110110

Key K1: 11110101

Key K2: 01100011

Encryption Process Starts........

After Initial Permutation(IP) : 01111001

before Swap : 00001001

After Swap : 10010000

before IP inverse : 10000000

Encrypted Message: 01000000

Decryption Process Starts........

01000000

After Permutation : 10000000

before Swap : 10010000

After Swap : 00001001

Before Extraction Permutation : 1001

After Extraction Permutation : 10110110

Decrypted Message: 10110110