**EX.NO.4: DATA ENCRYPTION STANDARD (DES)**

**CODE:**

**DES.java:**

import java.util.\*;

class DesData {

String plainText = new String();

String cipherText = new String();

String key = new String();

String keys[] = new String[16];

}

class DoubleDes {

String plainText = new String();

String cipherText = new String();

String key1 = new String();

String key2 = new String();

String keys1[] = new String[16];

String keys2[] = new String[16];

}

class TripleDes {

String plainText = new String();

String cipherText = new String();

String key1 = new String();

String key2 = new String();

String key3 = new String();

String keys1[] = new String[16];

String keys2[] = new String[16];

String keys3[] = new String[16];

}

class DES {

static Scanner sc = new Scanner(System.in);

String leftCircularShift(String input, int numBits) {

int n = input.length();

String shifted = "";

shifted += input.substring(numBits, n);

shifted += input.substring(0, numBits);

return shifted;

}

String permutation(int[] table, String input) {

String output = "";

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

output += input.charAt(table[i] - 1);

}

return output;

}

String[] generateKeys(String key) {

String keys[] = new String[16];

key = Conversion.asciiToBinary(key);

key = permutation(Constants.PC1, key);

System.out.println(

"\nOutput of PC1 (56-bit) in hex is " +

Conversion.binaryToHex(key).toUpperCase()

);

String roundKey = key;

System.out.println("\nThe round keys (48-bit) in hex are: ");

for (int i = 0; i < 16; i++) {

roundKey =

leftCircularShift(roundKey.substring(0, 28), Constants.shiftBits[i]) +

leftCircularShift(roundKey.substring(28, 56), Constants.shiftBits[i]);

keys[i] = permutation(Constants.PC2, roundKey);

System.out.println(

"Key " + (i + 1) + ": " + Conversion.binaryToHex(keys[i]).toUpperCase()

);

}

return keys;

}

String xor(String a, String b) {

String ans = "";

int n = a.length();

for (int i = 0; i < n; i++) {

if (a.charAt(i) == b.charAt(i)) ans += "0"; else ans += "1";

}

return ans;

}

String sBox(String input) {

String output = "";

String bin = "";

for (int i = 0; i < 48; i += 6) {

String temp = input.substring(i, i + 6);

int num = i / 6;

int row = Conversion.binaryToDecimal(

temp.charAt(0) + "" + temp.charAt(5)

);

int col = Conversion.binaryToDecimal(temp.substring(1, 5));

bin = Integer.toBinaryString(Constants.SBOX[num][row][col]);

if (bin.length() < 4) {

int l = 4 - bin.length();

for (int j = 0; j < l; j++) bin = '0' + bin;

}

output += bin;

}

return output;

}

String round(String input, String key, int r\_num) {

String left = input.substring(0, 32);

String right = input.substring(32, 64);

String temp = right;

// Expansion permutation 32 to 48 bit

temp = permutation(Constants.EP, temp);

// xor temp and round key

temp = xor(temp, key);

// lookup in s-box table

temp = sBox(temp);

// Straight D-box

temp = permutation(Constants.P, temp);

// xor

left = xor(left, temp);

// swapping

return right + left;

}

void tripleEncrypt() {

System.out.println("\nTRIPLE DES ENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

TripleDes des3Data = new TripleDes();

System.out.print("\nEnter the plainText of (in ASCII): ");

des3Data.plainText = sc.nextLine();

des3Data.plainText = Conversion.asciiToBinary(des3Data.plainText);

if (des3Data.plainText.length() % Constants.PT\_LENGTH != 0) {

des3Data.plainText = stuffPlainText(des3Data.plainText);

System.out.println(

"\nPlain text after bit stuffing (in hex) : " +

Conversion.binaryToHex(des3Data.plainText).toUpperCase() +

"\n"

);

}

System.out.print("\nEnter the first key of length 8 (in ASCII): ");

des3Data.key1 = sc.next();

sc.nextLine();

while (!validateKey(des3Data.key1)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key1 = sc.next();

sc.nextLine();

}

des3Data.keys1 = generateKeys(des3Data.key1);

des3Data.cipherText =

Conversion.binaryToHex(DesEncrypt(des3Data.plainText, des3Data.keys1));

System.out.println(

"\nThe intermediate cipher text is (in hex): " +

des3Data.cipherText.toUpperCase()

);

System.out.print("\nEnter the second key of length 8 (in ASCII): ");

des3Data.key2 = sc.next();

sc.nextLine();

while (!validateKey(des3Data.key2)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key2 = sc.next();

sc.nextLine();

}

des3Data.keys2 = generateKeys(des3Data.key2);

des3Data.cipherText = DesDecrypt(des3Data.cipherText, des3Data.keys2);

System.out.println(

"\nThe intermediate reverse cipher text is (in hex): " +

Conversion.binaryToHex(des3Data.cipherText).toUpperCase()

);

System.out.print("\nEnter the third key of length 8 (in ASCII): ");

des3Data.key3 = sc.next();

while (!validateKey(des3Data.key3)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key3 = sc.next();

}

des3Data.keys3 = generateKeys(des3Data.key3);

des3Data.cipherText =

Conversion.binaryToHex(DesEncrypt(des3Data.cipherText, des3Data.keys3));

System.out.println(

"\nThe cipher text is (in hex): " + des3Data.cipherText.toUpperCase()

);

}

void tripleDecrypt() {

System.out.println("\nTRIPLE DES DECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

TripleDes des3Data = new TripleDes();

System.out.print("\nEnter the cipherText (in hex): ");

des3Data.cipherText = sc.next();

sc.nextLine();

while (!validateCipherText(des3Data.cipherText)) {

System.out.println("\nInvalid cipher text length");

System.out.print("\nEnter the cipherText (in hex): ");

des3Data.cipherText = sc.next();

sc.nextLine();

}

System.out.print("\nEnter the key of length 8 (in ASCII): ");

des3Data.key1 = sc.next();

sc.nextLine();

while (!validateKey(des3Data.key1)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key1 = sc.next();

sc.nextLine();

}

des3Data.keys1 = generateKeys(des3Data.key1);

des3Data.plainText = DesDecrypt(des3Data.cipherText, des3Data.keys1);

System.out.println(

"\nThe intermediate plain text is (in hex): " +

Conversion.binaryToHex(des3Data.plainText).toUpperCase()

);

System.out.print("\nEnter the second key of length 8 (in ASCII): ");

des3Data.key2 = sc.next();

sc.nextLine();

while (!validateKey(des3Data.key2)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key2 = sc.next();

sc.nextLine();

}

des3Data.keys2 = generateKeys(des3Data.key2);

des3Data.plainText =

Conversion.binaryToHex(DesEncrypt(des3Data.plainText, des3Data.keys2));

System.out.println(

"\nThe intermediate reverse plain text is (in hex): " +

des3Data.plainText.toUpperCase()

);

System.out.print("\nEnter the third key of length 8 (in ASCII): ");

des3Data.key3 = sc.next();

sc.nextLine();

while (!validateKey(des3Data.key3)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des3Data.key3 = sc.next();

sc.nextLine();

}

des3Data.keys3 = generateKeys(des3Data.key3);

des3Data.plainText =

Conversion.binaryToAscii(DesDecrypt(des3Data.plainText, des3Data.keys3));

System.out.println(

"\nThe plain text is (in ASCII): " + des3Data.plainText.toUpperCase()

);

}

void doubleEncrypt() {

System.out.println("\nDOUBLE DES ENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

DoubleDes des2Data = new DoubleDes();

System.out.print("\nEnter the plainText of (in ASCII): ");

des2Data.plainText = sc.nextLine();

des2Data.plainText = Conversion.asciiToBinary(des2Data.plainText);

if (des2Data.plainText.length() % Constants.PT\_LENGTH != 0) {

des2Data.plainText = stuffPlainText(des2Data.plainText);

System.out.println(

"\nPlain text after bit stuffing (in hex) : " +

Conversion.binaryToHex(des2Data.plainText).toUpperCase() +

"\n"

);

}

System.out.print("\nEnter the first key of length 8 (in ASCII): ");

des2Data.key1 = sc.next();

sc.nextLine();

while (!validateKey(des2Data.key1)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des2Data.key1 = sc.next();

sc.nextLine();

}

des2Data.keys1 = generateKeys(des2Data.key1);

des2Data.cipherText = DesEncrypt(des2Data.plainText, des2Data.keys1);

System.out.println(

"\nThe intermediate cipher text is (in hex): " +

Conversion.binaryToHex(des2Data.cipherText).toUpperCase()

);

System.out.print("\nEnter the second key of length 8 (in ASCII): ");

des2Data.key2 = sc.next();

sc.nextLine();

while (!validateKey(des2Data.key2)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des2Data.key2 = sc.next();

sc.nextLine();

}

des2Data.keys2 = generateKeys(des2Data.key2);

des2Data.cipherText =

Conversion.binaryToHex(DesEncrypt(des2Data.cipherText, des2Data.keys2));

System.out.println(

"\nThe cipher text is (in hex): " + des2Data.cipherText.toUpperCase()

);

}

void doubleDecrypt() {

System.out.println("\nDOUBLE DES DECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

DoubleDes des2Data = new DoubleDes();

System.out.print("\nEnter the cipherText (in hex): ");

des2Data.cipherText = sc.next();

sc.nextLine();

while (!validateCipherText(des2Data.cipherText)) {

System.out.println("\nInvalid cipher text length");

System.out.print("\nEnter the cipherText (in hex): ");

des2Data.cipherText = sc.next();

sc.nextLine();

}

System.out.print("\nEnter the key of length 8 (in ASCII): ");

des2Data.key1 = sc.next();

sc.nextLine();

while (!validateKey(des2Data.key1)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des2Data.key1 = sc.next();

sc.nextLine();

}

des2Data.keys1 = generateKeys(des2Data.key1);

des2Data.plainText =

Conversion.binaryToHex(DesDecrypt(des2Data.cipherText, des2Data.keys1));

System.out.println(

"\nThe intermediate plain text is (in hex): " +

des2Data.plainText.toUpperCase()

);

System.out.print("\nEnter the second key of length 8 (in ASCII): ");

des2Data.key2 = sc.next();

sc.nextLine();

while (!validateKey(des2Data.key2)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

des2Data.key2 = sc.next();

sc.nextLine();

}

des2Data.keys2 = generateKeys(des2Data.key2);

des2Data.plainText =

Conversion.binaryToAscii(DesDecrypt(des2Data.plainText, des2Data.keys2));

System.out.println(

"\nThe plain text is (in ASCII): " + des2Data.plainText.toUpperCase()

);

}

void encrypt() {

System.out.println("\nENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

DesData desData = new DesData();

System.out.print("\nEnter the plainText of (in ASCII): ");

desData.plainText = sc.nextLine();

desData.plainText = Conversion.asciiToBinary(desData.plainText);

if (desData.plainText.length() % Constants.PT\_LENGTH != 0) {

desData.plainText = stuffPlainText(desData.plainText);

System.out.println(

"\nPlain text after bit stuffing (in hex) : " +

Conversion.binaryToHex(desData.plainText).toUpperCase() +

"\n"

);

}

System.out.print("\nEnter the key of length 8 (in ASCII): ");

desData.key = sc.next();

sc.nextLine();

while (!validateKey(desData.key)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

desData.key = sc.next();

sc.nextLine();

}

desData.keys = generateKeys(desData.key);

desData.cipherText =

Conversion.binaryToHex(DesEncrypt(desData.plainText, desData.keys));

System.out.println(

"\nThe cipher text is (in hex): " + desData.cipherText.toUpperCase()

);

}

String DesEncrypt(String plainText, String keys[]) { //pt,ct and keys in binary

String cipher = "";

int k = plainText.length();

String cipherBlock, initial\_perm = "";

for (int i = 0; i < k; i = i + Constants.PT\_LENGTH) {

cipherBlock = plainText.substring(i, i + Constants.PT\_LENGTH);

// initial permutation

cipherBlock = permutation(Constants.IP, cipherBlock);

initial\_perm += Conversion.binaryToHex(cipherBlock).toUpperCase();

// 16 rounds

for (int j = 0; j < 16; j++) {

cipherBlock = round(cipherBlock, keys[j], j);

}

// 32-bit swap

cipherBlock =

cipherBlock.substring(32, 64) + cipherBlock.substring(0, 32);

// final permutation

cipherBlock = permutation(Constants.FP, cipherBlock);

//cipherBlock = Conversion.binaryToHex(cipherBlock);

cipher += cipherBlock;

}

return cipher;

}

void decrypt() {

System.out.println("\nDECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

DesData desData = new DesData();

System.out.print("\nEnter the cipherText (in hex): ");

desData.cipherText = sc.next();

sc.nextLine();

while (!validateCipherText(desData.cipherText)) {

System.out.println("\nInvalid cipher text length");

System.out.print("\nEnter the cipherText (in hex): ");

desData.cipherText = sc.next();

sc.nextLine();

}

System.out.print("\nEnter the key of length 8 (in ASCII): ");

desData.key = sc.next();

sc.nextLine();

while (!validateKey(desData.key)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

desData.key = sc.next();

sc.nextLine();

}

desData.keys = generateKeys(desData.key);

desData.plainText =

Conversion.binaryToAscii(DesDecrypt(desData.cipherText, desData.keys));

System.out.println(

"\nThe plain text is (in ASCII): " + desData.plainText.toUpperCase()

);

}

String DesDecrypt(String cipherText, String keys[]) { //ct in hex, keys,pt in binary

String plain = "";

int k = cipherText.length();

String plainBlock, initial\_perm = "";

for (int i = 0; i < k; i = i + Constants.CT\_LENGTH) {

plainBlock = cipherText.substring(i, i + Constants.CT\_LENGTH);

plainBlock = Conversion.hexToBinary(plainBlock);

//initial permutation

plainBlock = permutation(Constants.IP, plainBlock);

// 16-rounds

for (int j = 15; j > -1; j--) {

plainBlock = round(plainBlock, keys[j], 15 - j);

}

// 32-bit swap

plainBlock = plainBlock.substring(32, 64) + plainBlock.substring(0, 32);

plainBlock = permutation(Constants.FP, plainBlock);

//plainBlock = Conversion.binaryToAscii(plainBlock);

plain += plainBlock;

}

return plain;

}

boolean validateKey(String key) {

return key.length() == Constants.KEY\_LENGTH;

}

String stuffPlainText(String plainText) {

int l = plainText.length();

if (l < Constants.PT\_LENGTH) {

for (int j = 0; j < Constants.PT\_LENGTH - l; j++) plainText =

plainText + '0';

} else {

for (

int j = 0;

j < Constants.PT\_LENGTH - (l % Constants.PT\_LENGTH);

j++

) plainText = plainText + '0';

}

return plainText;

}

boolean validateCipherText(String str) {

return (str.length() % Constants.CT\_LENGTH == 0);

}

public static void main(String args[]) {

DES des = new DES();

int choice = 0;

while (true) {

System.out.println("\nDATA ENCRYPTION STANDARD - DES");

System.out.println("------------------------------");

System.out.println("\n1.Key Generation");

System.out.println("\n2.Encryption");

System.out.println("\n3.Decryption");

System.out.println("\n4.Double DES");

System.out.println("\n5.Triple DES");

System.out.println("\n6.Exit");

System.out.print("\nEnter your choice(1/2/3/4/5/6): ");

choice = DES.sc.nextInt();

sc.nextLine();

if (choice == 1) {

System.out.println("\nKEY - GENERATION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter the key of length 8 (in ASCII): ");

String key = sc.next();

while (!des.validateKey(key)) {

System.out.println("\nInvalid key");

System.out.print("\nEnter the key: ");

key = sc.next();

}

String keys[] = des.generateKeys(key);

} else if (choice == 2) {

des.encrypt();

} else if (choice == 3) {

des.decrypt();

} else if (choice == 4) {

System.out.println("\nDouble DES");

System.out.println("------------");

des.doubleEncrypt();

des.doubleDecrypt();

} else if (choice == 5) {

System.out.println("\nTriple DES");

System.out.println("------------");

des.tripleEncrypt();

des.tripleDecrypt();

} else {

break;

}

}

sc.close();

}

}

**Constants.java**

public final class Constants {

public static final int KEY\_LENGTH=8; //in ascii

public static final int PT\_LENGTH=64; //in binary

public static final int CT\_LENGTH=16; //in hex

// Initial Permutation Table

public static final int[] IP = { 58, 50, 42, 34, 26, 18,

10, 2, 60, 52, 44, 36, 28, 20,

12, 4, 62, 54, 46, 38,

30, 22, 14, 6, 64, 56,

48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17,

9, 1, 59, 51, 43, 35, 27,

19, 11, 3, 61, 53, 45,

37, 29, 21, 13, 5, 63, 55,

47, 39, 31, 23, 15, 7 };

// Inverse Initial Permutation Table

public static final int[] FP = { 40, 8, 48, 16, 56, 24, 64,

32, 39, 7, 47, 15, 55,

23, 63, 31, 38, 6, 46,

14, 54, 22, 62, 30, 37,

5, 45, 13, 53, 21, 61,

29, 36, 4, 44, 12, 52,

20, 60, 28, 35, 3, 43,

11, 51, 19, 59, 27, 34,

2, 42, 10, 50, 18, 58,

26, 33, 1, 41, 9, 49,

17, 57, 25 };

// first key-Permutation Table

public static final int[] PC1 = { 57, 49, 41, 33, 25,

17, 9, 1, 58, 50, 42, 34, 26,

18, 10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36, 63,

55, 47, 39, 31, 23, 15, 7, 62,

54, 46, 38, 30, 22, 14, 6, 61,

53, 45, 37, 29, 21, 13, 5, 28,

20, 12, 4 };

// second key-Permutation Table

public static final int[] PC2 = { 14, 17, 11, 24, 1, 5, 3,

28, 15, 6, 21, 10, 23, 19, 12,

4, 26, 8, 16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32 };

// Expansion D-box Table

public static final int[] EP = { 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 };

// Straight Permutation Table

public static final int[] P = { 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 };

// S-box Table

public static final int[][][] SBOX = {

{ { 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 } }

};

public static final int[] shiftBits = { 1, 1, 2, 2, 2, 2, 2, 2,

1, 2, 2, 2, 2, 2, 2, 1 };

}

**Conversion.java:**

public final class Conversion {

public static final String asciiToBinary(String ascii) {

String bin\_T = "";

int n = ascii.length();

for (int i = 0; i < n; i++) {

int val = Integer.valueOf(ascii.charAt(i));

String bin = "";

while (val > 0) {

if (val % 2 == 1) bin += '1'; else bin += '0';

val /= 2;

}

bin = reverse(bin);

if (bin.length() < 8) {

int l = 8 - bin.length();

for (int j = 0; j < l; j++) bin = '0' + bin;

}

bin\_T += bin + "";

}

return bin\_T;

}

public static final String reverse(String input) {

char[] a = input.toCharArray();

int l, r = 0;

r = a.length - 1;

for (l = 0; l < r; l++, r--) {

char temp = a[l];

a[l] = a[r];

a[r] = temp;

}

return String.valueOf(a);

}

public static final int binaryToDecimal(String binaryStr) {

return Integer.parseInt(binaryStr,2);

}

public static final String binaryToHex(String binary) {

String hexStr = "";

for (int i = 0; i < binary.length(); i = i + 4) {

int decimal = Integer.parseInt(binary.substring(i, i + 4), 2);

hexStr = hexStr + Integer.toString(decimal, 16);

//Integer.toHexString()

}

return hexStr;

}

public static final String hexToBinary(String hex) {

String bin = "", temp = "";

int n = Integer.parseInt(hex.substring(0, 7), 16);

bin = Integer.toBinaryString(n);

if (bin.length() < 28) {

int l = 28 - bin.length();

for (int j = 0; j < l; j++) bin = '0' + bin;

}

n = Integer.parseInt(hex.substring(7, 14), 16);

temp = Integer.toBinaryString(n);

if (temp.length() < 28) {

int l = 28 - temp.length();

for (int j = 0; j < l; j++) temp = '0' + temp;

}

bin += temp;

n = Integer.parseInt(hex.substring(14, 16), 16);

temp = Integer.toBinaryString(n);

if (temp.length() < 8) {

int l = 8 - temp.length();

for (int j = 0; j < l; j++) temp = '0' + temp;

}

bin += temp;

return bin;

}

public static final String binaryToAscii(String binary) {

String res = "";

for (int i = 0; i < binary.length(); i = i + 8) {

char ch = (char) binaryToDecimal(binary.substring(i, i + 8));

res += ch;

}

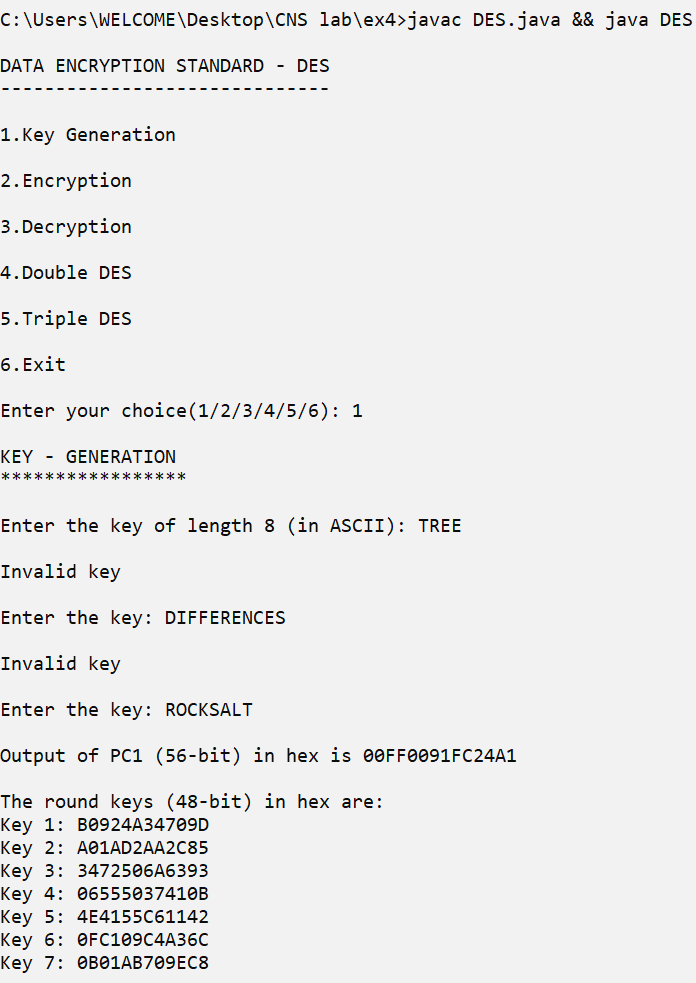
return res;

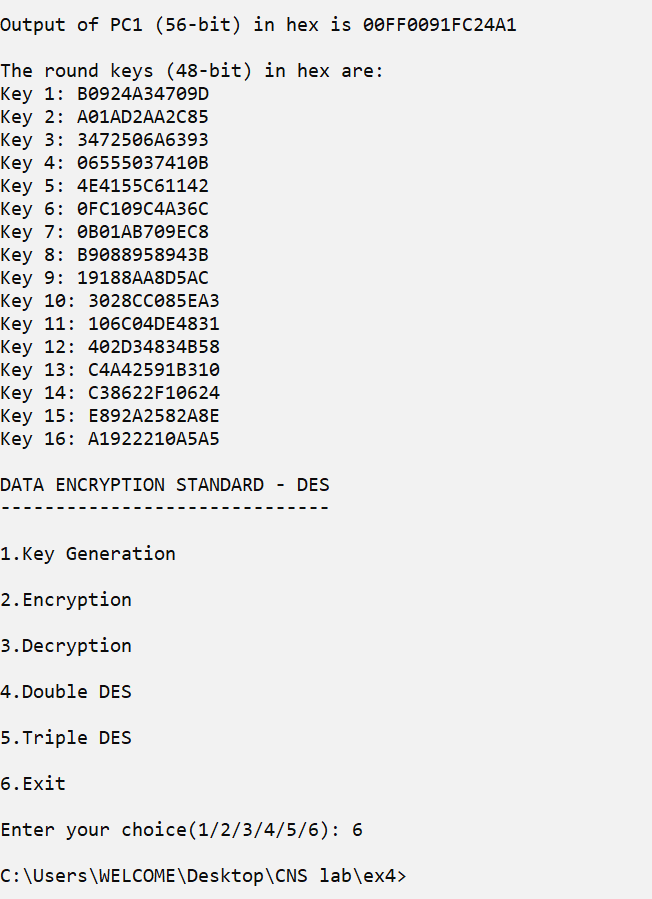
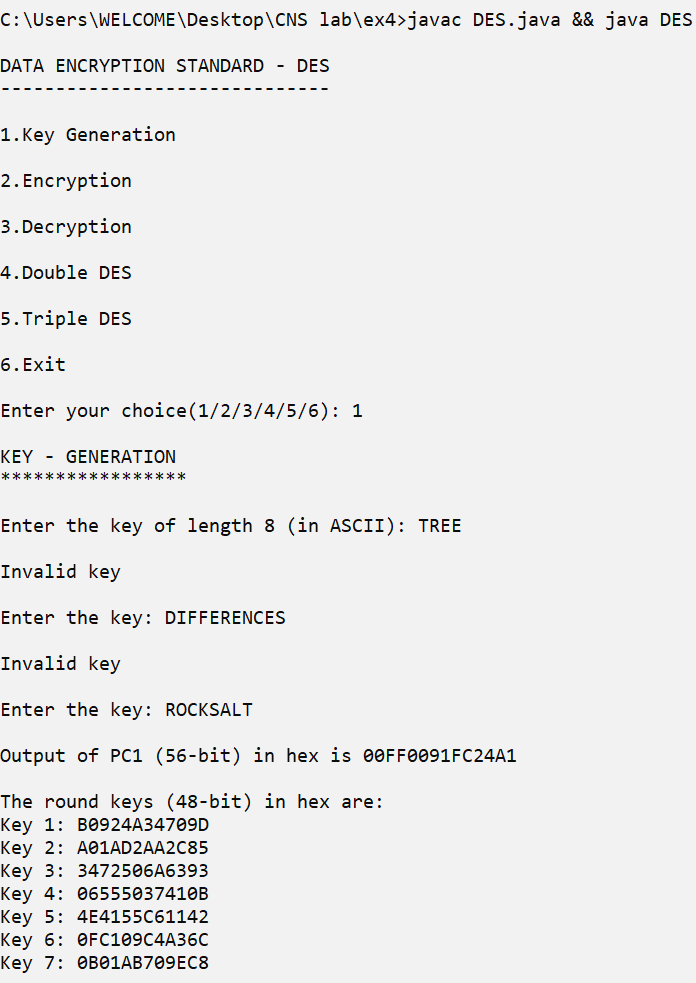
}

}

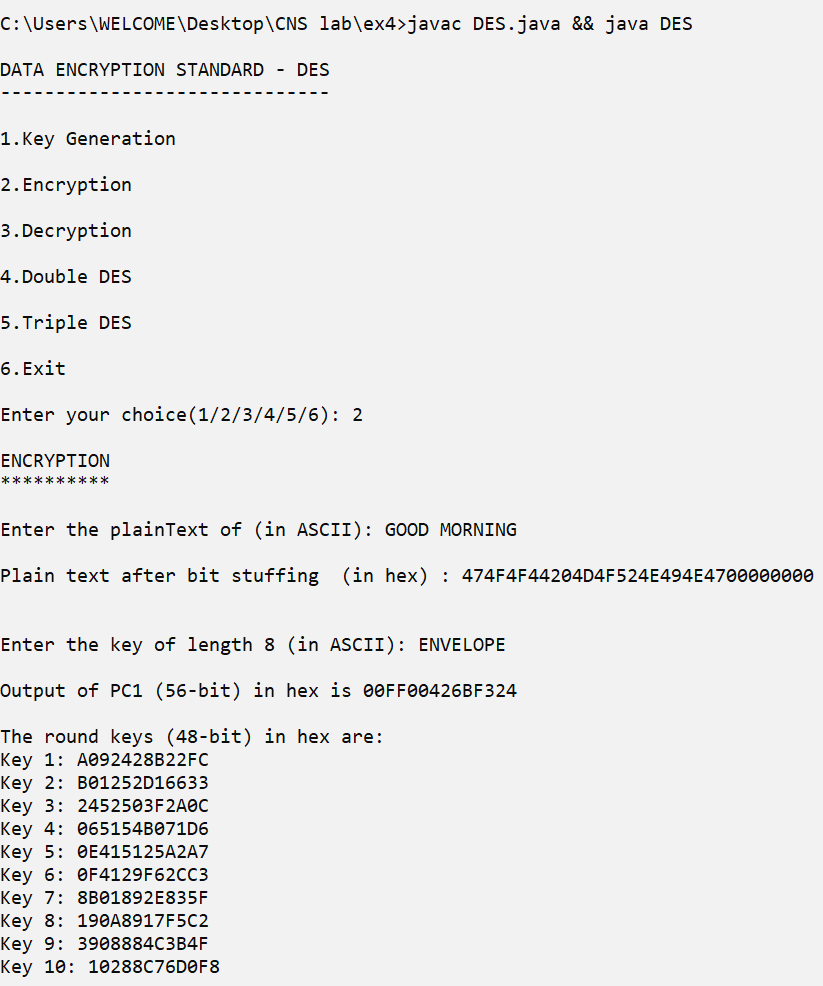
**OUTPUT:**

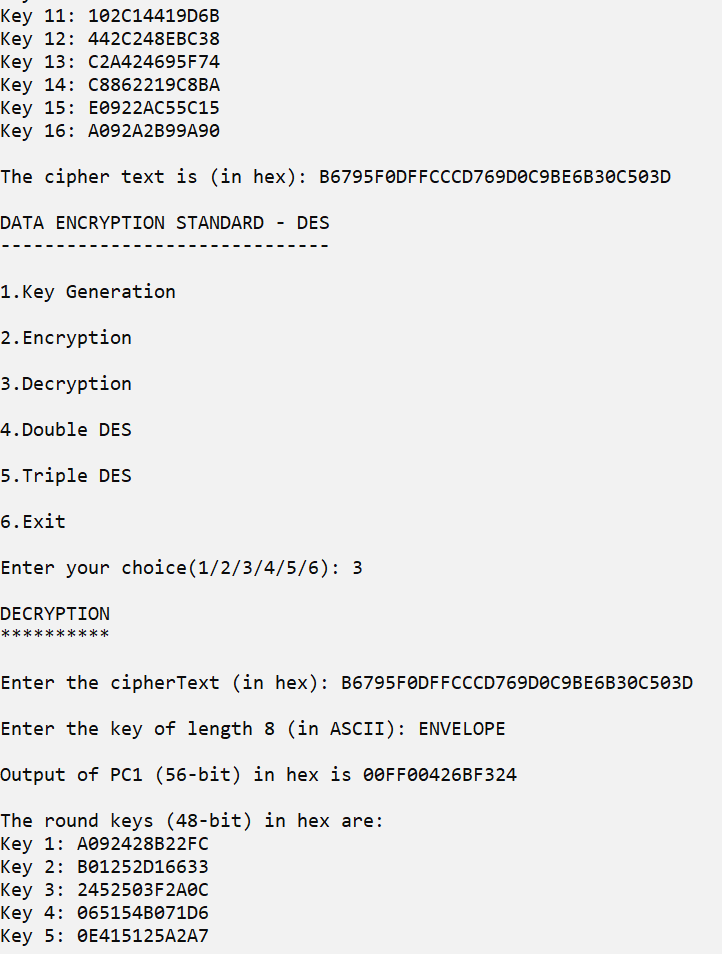
1. **Key Generation:**

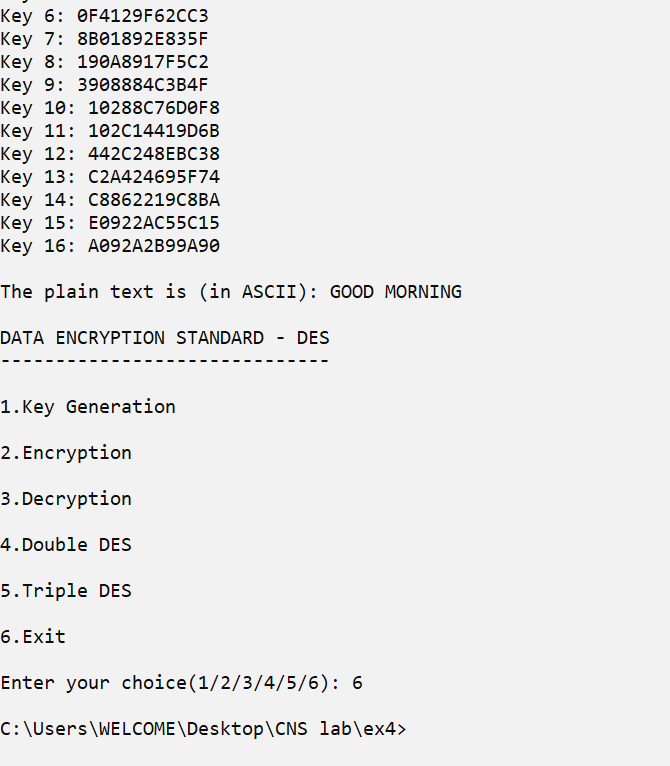




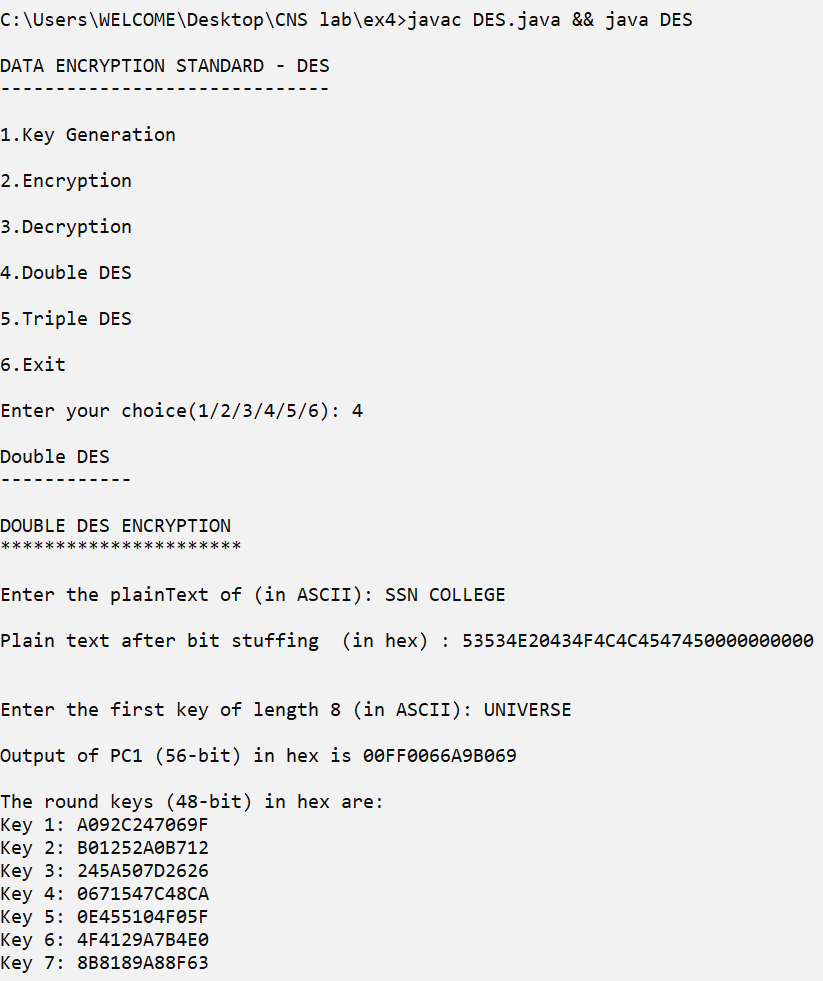
**2. DES ENCRYPTION & DECRYPTION:**

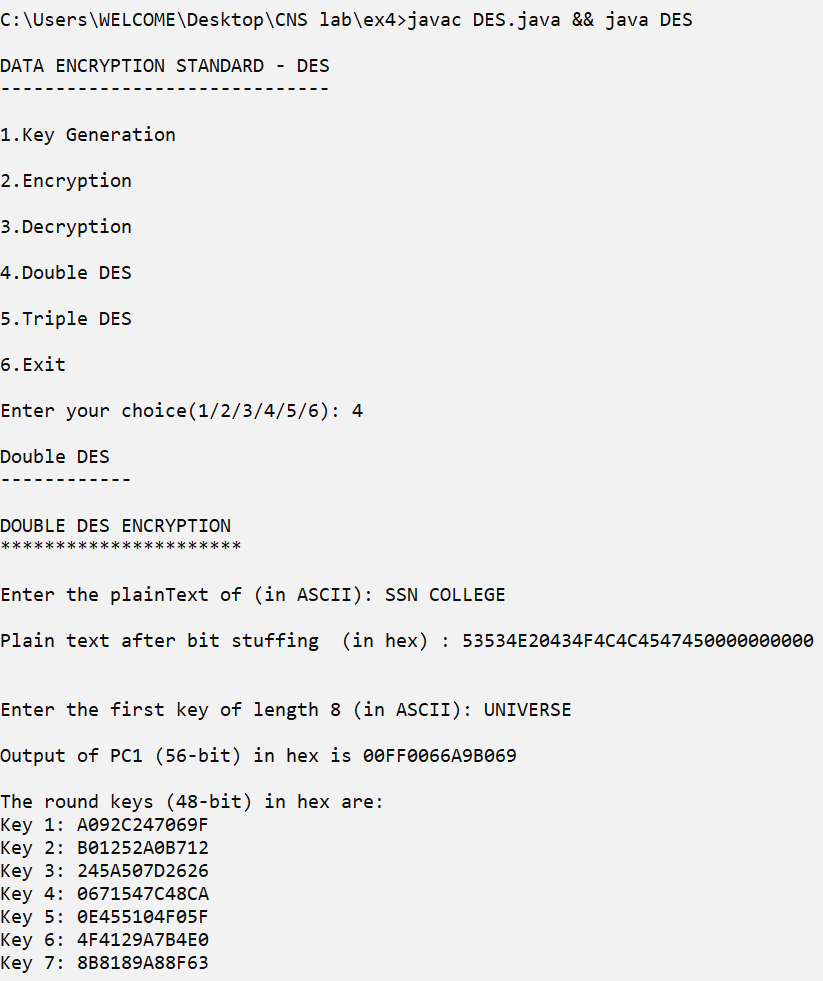


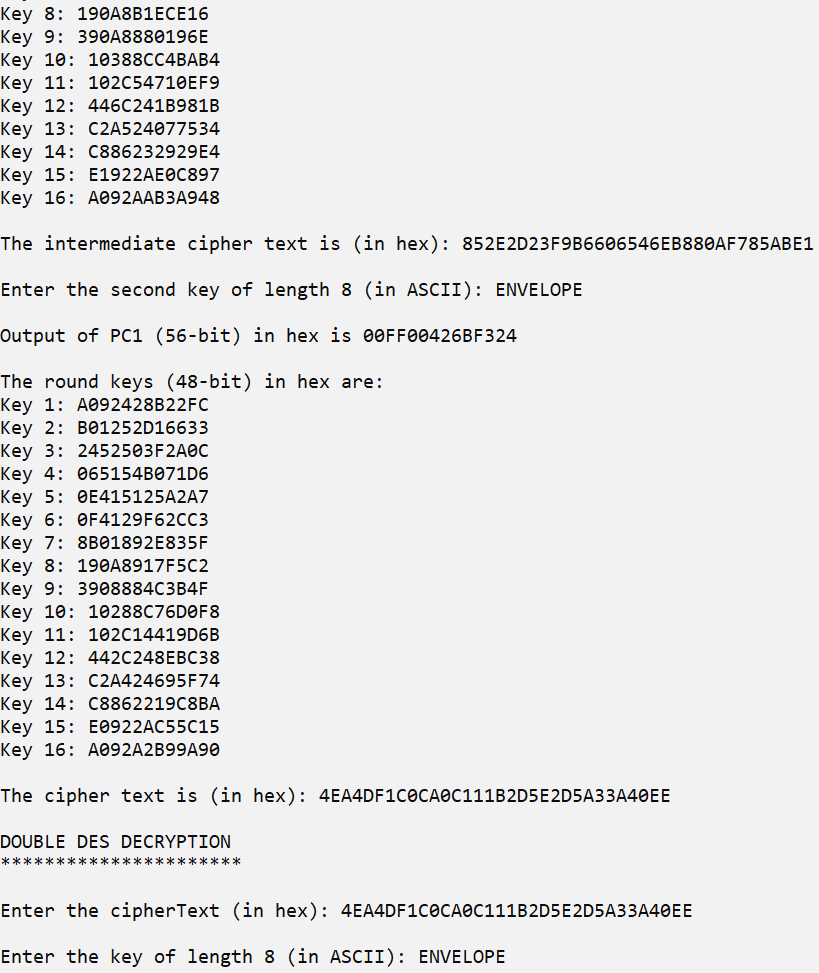


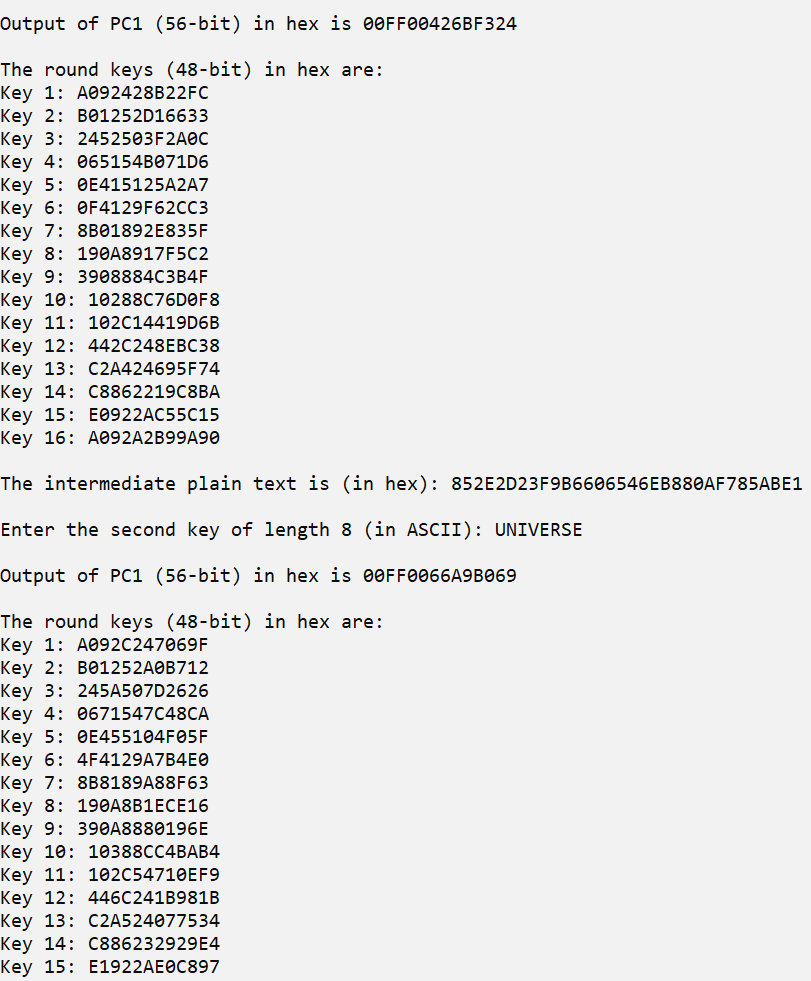
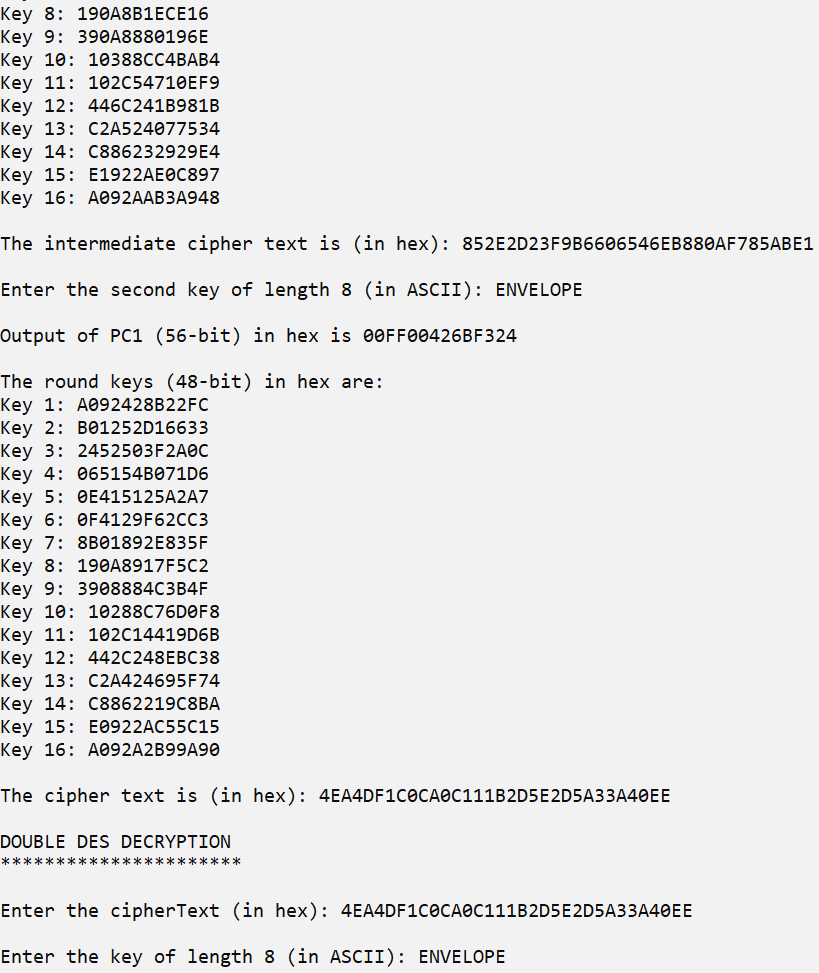
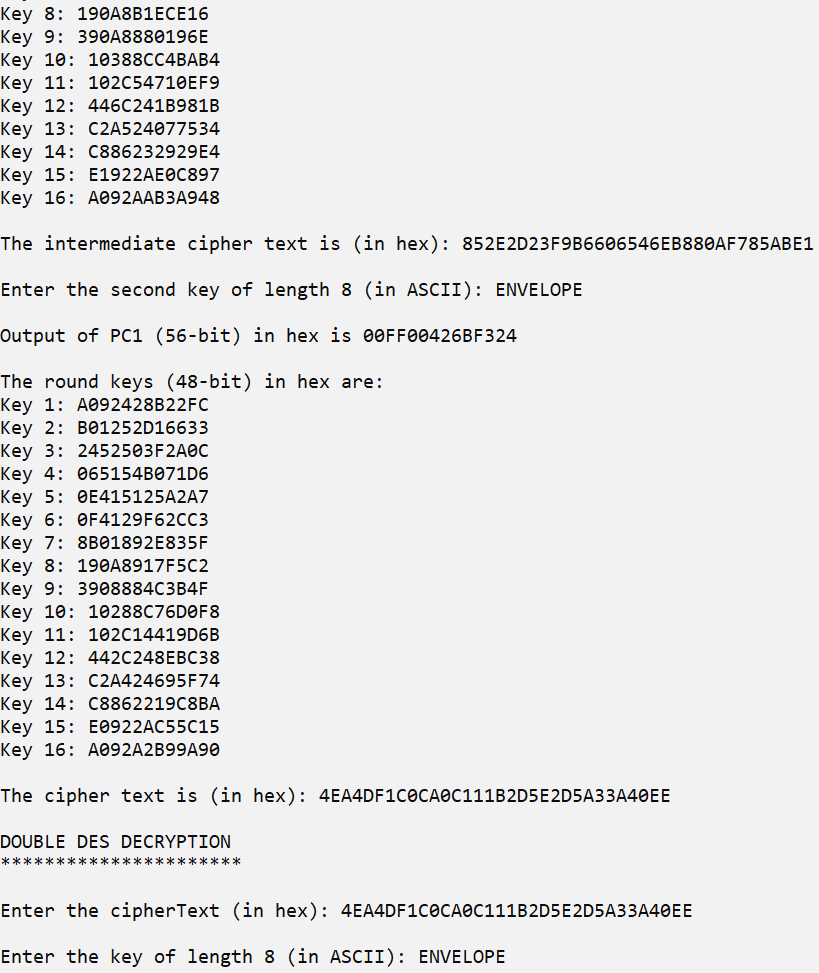


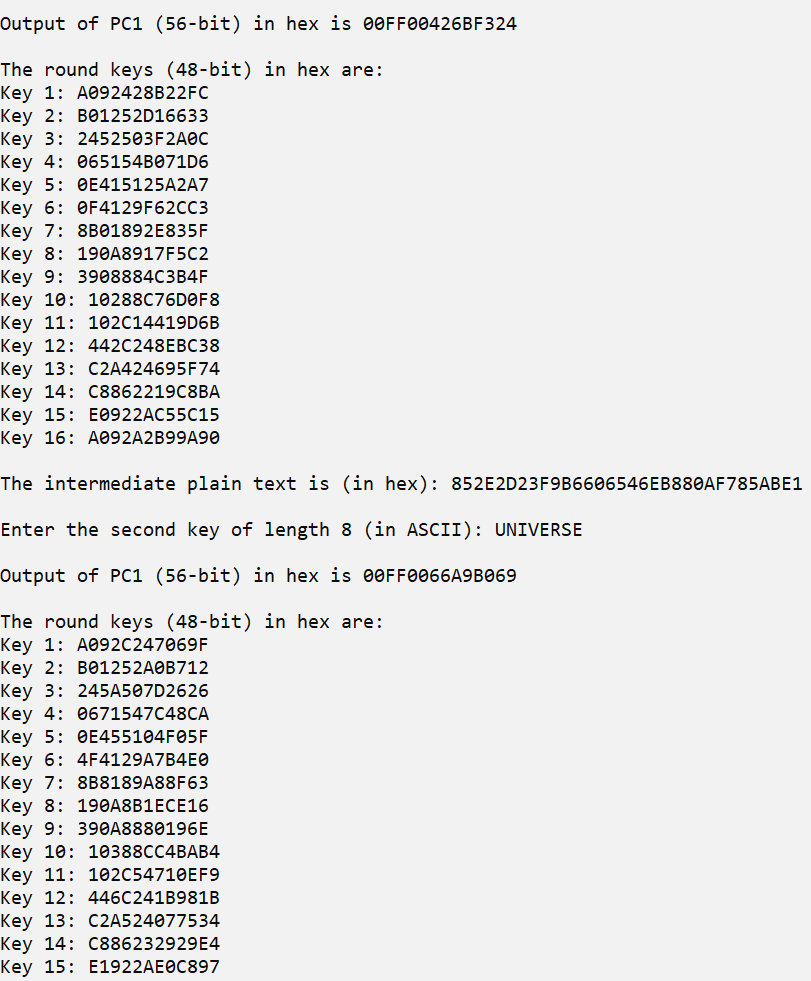
**3.DOUBLE DES**

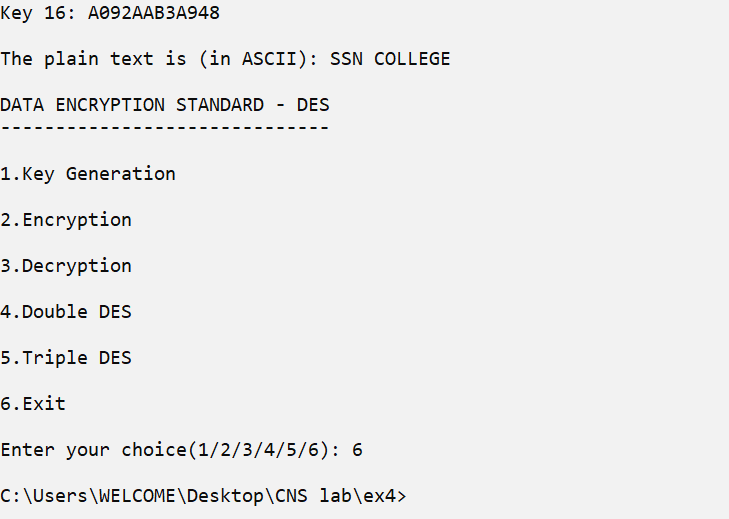
****

****

****

****

****

****

**4.TRIPLE DES:**

