Lab – 8 & 9

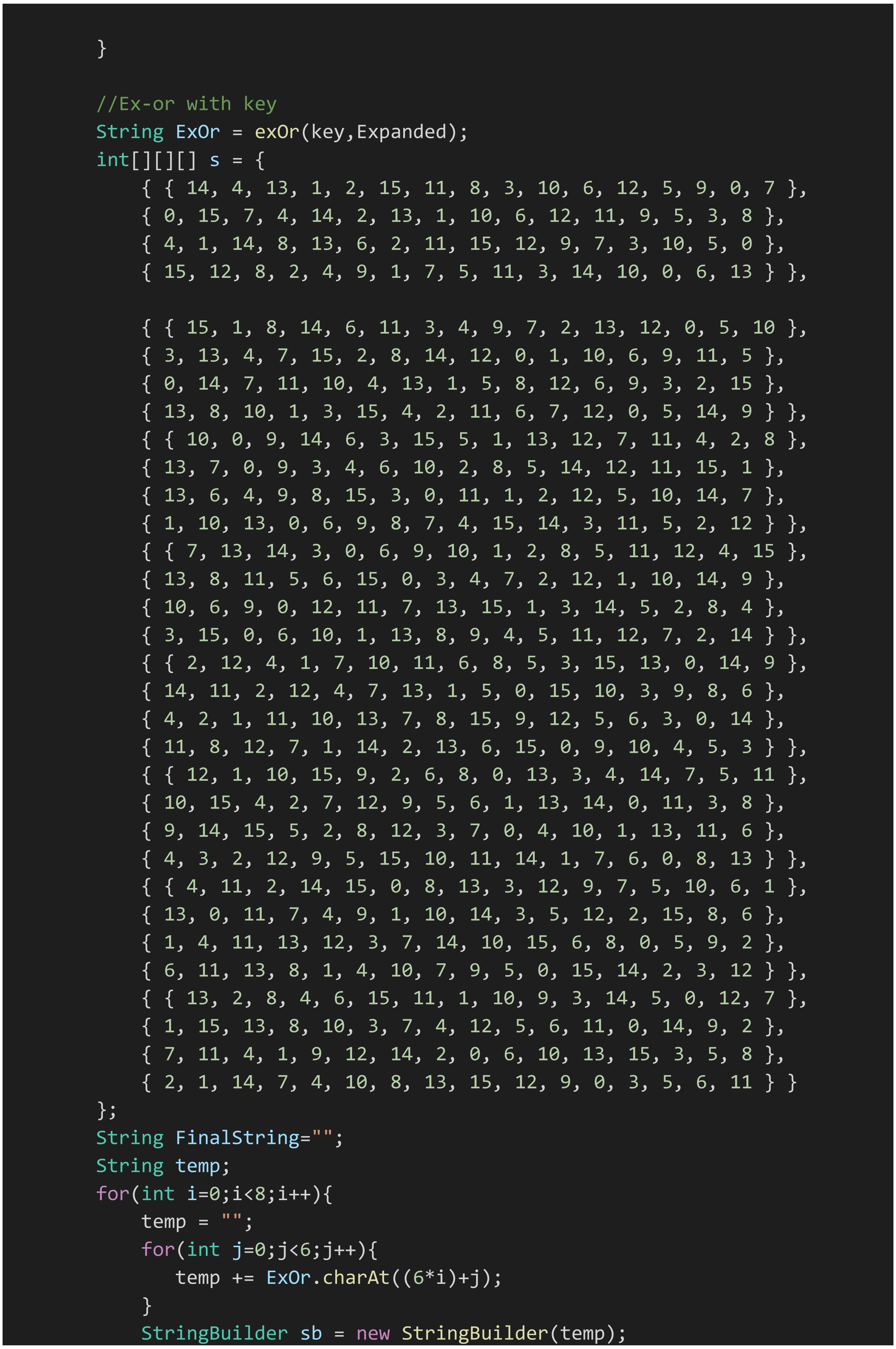
Subject: NIS

Aim: Implement DES (Data Encryption Standard) Encryption and Decryption algorithm with Key Generation.

Program: -

```java
import java.util.*;
import java.lang.*;
import java.math.BigInteger;

public class DES{

    public static String hexToBinary(String HEX){

        HashMap<Character,String> hashMap = new HashMap<Character,String>();
        String binary = "";

        hashMap.put('0', "0000");
        hashMap.put('1', "0001");
        hashMap.put('2', "0010");
        hashMap.put('3', "0011");
        hashMap.put('4', "0100");
        hashMap.put('5', "0101");
        hashMap.put('6', "0110");
        hashMap.put('7', "0111");
        hashMap.put('8', "1000");
        hashMap.put('9', "1001");
        hashMap.put('A', "1010");
        hashMap.put('B', "1011");
        hashMap.put('C', "1100");
        hashMap.put('D', "1101");
        hashMap.put('E', "1110");
        hashMap.put('F', "1111");

        char[] charArray = HEX.toCharArray();

        for(int i=0;i<HEX.length();i++){
            binary += hashMap.get(HEX.charAt(i));
        }

        return binary;
    }

    public static String binaryToHex(String binary){

        HashMap<String,Character> hashMap = new HashMap<String,Character>();
        String Hex = "";

        hashMap.put("0000",'0');
        hashMap.put("0001" ,'1');
        hashMap.put("0010" ,'2');
        hashMap.put("0011" ,'3');
        hashMap.put("0100" ,'4');
        hashMap.put("0101" ,'5');
        hashMap.put("0110" ,'6');
        hashMap.put("0111" ,'7');
        hashMap.put("1000" ,'8');
        hashMap.put("1001" ,'9');
        hashMap.put("1010" ,'A');
        hashMap.put("1011" ,'B');
        hashMap.put("1100" ,'C');
        hashMap.put("1101" ,'D');
        hashMap.put("1110" ,'E');
        hashMap.put("1111" ,'F');

        char[] charArray = binary.toCharArray();
        int f=0;
        for(int i=4;i<=binary.length();i=i+4){
            Hex += hashMap.get(binary.substring((i-4),i));
        }

        return Hex;
    }

    public static String[] keyGeneration(String key){

        //parity dropout
        String ParityDropout = parityDropout(key);
        //string division
        StringBuilder sb = new StringBuilder(ParityDropout);
        String left = sb.substring(0,ParityDropout.length()/2);
        String right = sb.substring(ParityDropout.length()/2,ParityDropout.len
gth());

        //circulershift
        String[] keys = new String[16];
        int count =1;
        for(int i=0;i<16;i++){

            if(count == 1 || count == 2 || count == 9 || count == 16 ){
                left = circulerShitLeft(left,true);
                right = circulerShitLeft(right,true);
                keys[i] = compressionFunction(left,right);
                count++;
            }
            else
            {
                left = circulerShitLeft(left,false);
                right = circulerShitLeft(right,false);
                keys[i] = compressionFunction(left,right);
                count++;

            }
        }
        return keys;

    }

    public static String initialPermutation(String PlainText){

        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};

        String IPermut = "";

        for(int i=0;i<IP.length;i++){
           IPermut += PlainText.charAt(IP[i]-1);
        }
        return IPermut;
    }

    public static String parityDropout(String key){
        int[] PD ={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};

        String dropout = "";
        for(int i=0;i<PD.length;i++){
            dropout += key.charAt(PD[i]-1);
        }
      return dropout;
    }

    public static String circulerShitLeft(String key,boolean flage){
        StringBuilder sb = new StringBuilder(key);
        String temp;
        if(flage==true){
             temp = sb.substring(1) + sb.substring(0,1);
```

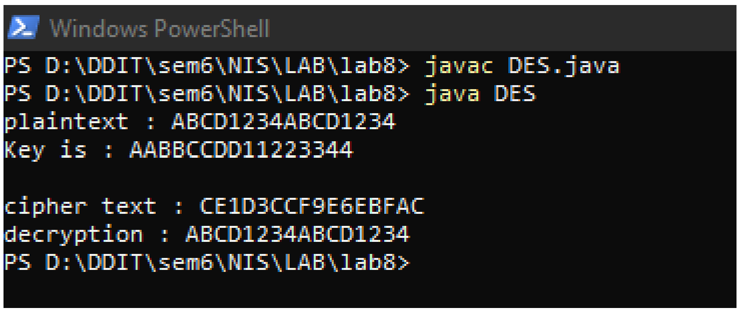
```java
        }else{
            temp = sb.substring(2) + sb.substring(0,2);
        }
        return temp;
    }

    public static String compressionFunction(String left,String right){

        int[] CompressionTable = {14, 17, 11, 24, 1, 5,3, 28, 15, 6, 21, 10, 2
3, 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};

        String compresed ="";
        String CompressionString =left + right;

        for(int i=0;i<CompressionTable.length;i++){
            compresed += CompressionString.charAt(CompressionTable[i]-1);
        }
        return compresed;
    }



    public static String exOr(String key,String Expanded){
        String ExOr = "";
        for(int i=0;i<key.length();i++){
            if(key.charAt(i)== Expanded.charAt(i)){
                ExOr += "0";
            }else{
                ExOr += "1";
            }
        }
        return ExOr;
    }

    public static String functionF(String HEX,String key){
        String Expanded = "";
        int[] ExpantionTable = { 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 };

        for(int i=0; i<ExpantionTable.length ;i++){
            Expanded += HEX.charAt(ExpantionTable[i]-1);
```



```java
    }

    //Ex-or with key
    String ExOr = exOr(key,Expanded);
    int[][][] s = {
      { { 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 } }
    };
    String FinalString="";
    String temp;
    for(int i=0;i<8;i++){
      temp = "";
      for(int j=0;j<6;j++){
         temp += ExOr.charAt((6*i)+j);
      }
      StringBuilder sb = new StringBuilder(temp);
```

```java
            int row = Integer.parseInt("" + temp.charAt(0) + temp.charAt(5) +
"",2);
            int column = Integer.parseInt(sb.substring(1,5),2);
            FinalString += Integer.toHexString(s[i][row][column]).toUpperCase(
);
        }

        FinalString = hexToBinary(FinalString);

        // Straight Permutation Table
        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 };

            String StraightForward="";
            for(int i=0; i<P.length ;i++){
                StraightForward += FinalString.charAt(P[i]-1);

            }
        return StraightForward;
    }

    public static String encryption(String PlainText,String[] keys){
        String IPermut = initialPermutation(PlainText);


        String  li_1 = IPermut.substring(0,32);
        String  ri_1= IPermut.substring(32);
        String left,right;
        for(int i=0;i<16;i++){
            left= ri_1;

            right= exOr(li_1,functionF(ri_1,keys[i]));


            li_1 = left;
            ri_1 = right;

        }
        String cipherText = li_1 + ri_1;
        cipherText = cipherText.substring(32,64)
                        + cipherText.substring(0, 32);

        String b = lastPermutation(cipherText);
        return b;
    }

    public static void decryption(String cipherText,String[] keys){
        String IPermut = initialPermutation(cipherText);


        String  li_1 = IPermut.substring(0,32);
        String  ri_1= IPermut.substring(32);
        String left,right;
        for(int i=15;i>=0;i--){
            left= ri_1;
            right= exOr(li_1,functionF(ri_1,keys[i]));
            li_1 = left;
            ri_1 = right;
        }
        String plainText = li_1 + ri_1;
        plainText = plainText.substring(32, 64)
                        + plainText.substring(0, 32);

        System.out.println("decryption : " + binaryToHex(lastPermutation(plain
Text)));

    }

    public static String lastPermutation(String IP_1){
        String temp="";

                int[] IP1 = {
                    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
                };
                    for(int i=0;i<IP1.length;i++){
                        temp += IP_1.charAt(IP1[i]-1);

                    }
                    return temp;
    }
    public static void main(String args[]){
        String PlainText = hexToBinary("ABCD1234ABCD1234");
        String key = hexToBinary("AABBCCDD11223344");
        System.out.println("plaintext : " +binaryToHex(PlainText));
        System.out.println("Key is : " + binaryToHex(key));
       String[] keys = keyGeneration(key);
        String ciphertext = encryption(PlainText,keys);
        System.out.println("\ncipher text : " + binaryToHex(ciphertext));
        decryption(ciphertext,keys);

    }
}
```

Output : -



Windows PowerShell

```
PS D:\DDIT\sem6\NIS\LAB\lab8> javac DES.java
PS D:\DDIT\sem6\NIS\LAB\lab8> java DES
plaintext : ABCD1234ABCD1234
Key is : AABBCCDD11223344

cipher text : CE1D3CCF9E6EBFAC
decryption : ABCD1234ABCD1234
PS D:\DDIT\sem6\NIS\LAB\lab8>
```