EX.NO.1

CAESAR CIPHER – ENCRYPTION, DECRYPTION AND CRYPTANALYSIS

CODE:

```
import java.util.Scanner;
import java.util.Arrays;
import java.util.HashSet;

public class CaesarCipher {

 public static String encrypt(String plaintext, int key) {
  String result = "";
  for (int i = 0; i < plaintext.length(); i++) {
   if (Character.isUpperCase(plaintext.charAt(i))) {
    char ch = (char) (((int) plaintext.charAt(i) - 65 + key) % 26 + 65);
    result += ch;
   } else {
    char ch = (char) (((int) plaintext.charAt(i) - 97 + key) % 26 + 97);
    result += ch;
   }
  }
  return result;
 }

 public static String decrypt(String ciphertext, int key) {
  String result = "";
  for (int i = 0; i < ciphertext.length(); i++) {
   if (Character.isUpperCase(ciphertext.charAt(i))) {
    char ch = (char) (((int) ciphertext.charAt(i) - 65 - key + 26) % 26 + 65);
    result += ch;
   } else {
    char ch = (char) (((int) ciphertext.charAt(i) - 97 - key + 26) % 26 + 97);
    result += ch;
   }
  }
  return result;
 }

 public static boolean validateString(String str) {
  str = str.toLowerCase();
  char[] charArray = str.toCharArray();
  for (int i = 0; i < charArray.length; i++) {
   char ch = charArray[i];
```

```
  if (!((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))) {
      return false;
    }
   }
   return true;
  }

  public static boolean validateKey(int key) {
   return (key >= 0 && key <= 25);
  }

  public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   String[] dictionary = { "hello", "zebra", "pen", "jug" };
   HashSet<String> dict = new HashSet(Arrays.asList(dictionary));
   boolean flag=false;

   System.out.print("\nEnter the Plain Text: ");
   String plaintext = sc.next();
   while (!validateString(plaintext)) {
    System.out.println("\nPlain text can contain only alphabets");
    System.out.print("\nEnter the Plain Text: ");
    plaintext = sc.next();
   }
   int key = -1;
   System.out.print("\nEnter encryption key: ");
   String k =sc.next();

   while(!(k.length()<=2 && k.length()>=1 && k.matches("[0-9][0-9]") &&
validateKey(Integer.parseInt(k)))){
    System.out.println("\nInvalid key");
    System.out.print("\nEnter encryption  key: ");
    k =sc.next();
   }

   key=Integer.parseInt(k);

   System.out.println("\nCipher text is : " + encrypt(plaintext, key));

   System.out.print("\nEnter the Cipher Text: ");
   String ciphertext = sc.next();
   while (!validateString(ciphertext)) {
    System.out.println("\nCipher text can only contain lower case alphabets");
    System.out.print("\nEnter the Cipher Text: ");
    ciphertext = sc.next();
   }
```

```
    key = -1;
    System.out.print("\nEnter decryption key: ");
    k =sc.next();

    while(!(k.length()<=2 && k.length()>=1 && k.matches("[0-9][0-9]") &&
validateKey(Integer.parseInt(k)))){
      System.out.println("\nInvalid key");
      System.out.print("\nEnter decryption  key: ");
      k =sc.next();
    }

    key=Integer.parseInt(k);

    System.out.println("\nPlain text is : " + decrypt(ciphertext, key));

    System.out.print("\nEnter the Cipher Text for crypt analysis: ");
    String crypt = sc.next();
    while (!validateString(crypt)) {
      System.out.println("\nCipher text can only contain lower case alphabets");
      System.out.print("\nEnter the Cipher Text for crypt analysis: ");
      crypt = sc.next();
    }
    System.out.print("Key  PlainText");
    System.out.print("\n***  *********");
    String values = "";
    for (int i = 0; i < 26; i++) {
      String res = decrypt(crypt, i);
      System.out.printf("\n%-3d  %s", i, res);
      if (dict.contains(res.toLowerCase())) {
        values += "Key = " + i + " : " + res + "\n";
        break;
      }
    }
    System.out.println("\nThe possible plain text value is: " + values);
  }
}
```

OUTPUT:

Example 1:

```
C:\Users\WELCOME\Desktop\CNS lab\ex1>java CaesarCipher

ENCRYPTION
**********

Enter the Plain Text: ABC123

Plain text can contain only alphabets

Enter the Plain Text: ZeBrA

Enter encryption key: 22

Cipher text is : VaXnW

DECRYPTION
**********

Enter the Cipher Text: vaxNW

Enter decryption key: 22

Plain text is : zebRA

CRYPT-ANALYSIS
**************

Enter the Cipher Text for crypt analysis: vaxnw
Key  PlainText
***  *********
0    vaxnw
1    uzwmv
2    tyvlu
3    sxukt
4    rwtjs
5    qvsir
6    purhq
7    otqgp
8    nspfo
9    mroen
10   lqndm
11   kpmcl
12   jolbk
13   inkaj
14   hmjzi
15   gliyh
16   fkhxg
17   ejgwf
18   difve
19   cheud
20   bgdtc
21   afcsb
22   zebra
The possible plain text value is: Key = 22 : zebra


C:\Users\WELCOME\Desktop\CNS lab\ex1>
```

Example 2:

```
C:\Users\WELCOME\Desktop\CNS lab\ex1>java CaesarCipher

ENCRYPTION
**********

Enter the Plain Text: lion

Enter encryption key: -3

Invalid key

Enter encryption  key: abcd

Invalid key

Enter encryption  key: 45

Invalid key

Enter encryption  key: 2

Cipher text is : nkqp

DECRYPTION
**********

Enter the Cipher Text: nkqp

Enter decryption key: 2

Plain text is : lion

CRYPT-ANALYSIS
**************

Enter the Cipher Text for crypt analysis: nkqp
Key  PlainText
***  *********
0    nkqp
1    mjpo
2    lion
The possible plain text value is: Key = 2 : lion
```

PLAYFAIR CIPHER – ENCRYPTION AND DECRYPTION

CODE:

```
import java.util.Scanner;

public class PlayFairCipher {

   String keyword = new String();
   String plainText = new String();
   char key_mat[][] = new char[5][5];
   String cipherText = new String();

   public void validateKey() {
      boolean isKeyValid = false;
      if (keyword.contains("j")) {
         isKeyValid = true;
         keyword = keyword.replace('j', 'i');
      }
      // remove duplicates
      String str = new String();
      for (int i = 0; i < keyword.length(); i++) {
         char c = keyword.charAt(i);
         if (str.indexOf(c) < 0)
            str += c;
      }
      if (isKeyValid || !keyword.equals(str)) {
         keyword = str;
         System.out.println("Modified key is ------ " + keyword);
      }
      // generating matrix entries as a string
      boolean flag = true;
      char current;
      char drop_char = 'j';
      for (int i = 0; i < 26; i++) {
         current = (char) (i + 97);
         if (current == drop_char)
            continue;
         for (int j = 0; j < keyword.length(); j++) {
            if (current == keyword.charAt(j)) {
               flag = false;
               break;
            }
         }
         if (flag)
```

```
            keyword = keyword + current;
        flag = true;
    }
    // System.out.println("key is " + keyword);
}

public void printKeyMatrix() {
    System.out.println("\nThe key matrix is\n");
    int idx = 0;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            key_mat[i][j] = keyword.charAt(idx);
            System.out.print(key_mat[i][j] + " ");
            idx++;
        }
        System.out.println();
    }
}

public void modifyPlainText() {
    if (plainText.contains("j")) {
        plainText = plainText.replace('j', 'i');
        System.out.println("After replacing j with i in the plain text ---- " + plainText);
    }
    StringBuffer newString = new StringBuffer(plainText);

    for (int i = 0; i < newString.length() - 1; i++) {
        // check for reptition in a pair
        if (i % 2 == 0) {
            if (newString.charAt(i) == newString.charAt(i + 1)) {
                if (newString.charAt(i) != 'z')
                    newString.insert(i + 1, "z");
                else
                    newString.insert(i + 1, "q");
            }

        }
    }
    if (newString.length() % 2 != 0) {
        if (newString.charAt(newString.length() - 1) != 'z')
            newString.append("z");
        else
            newString.append("q");
    }

    if (!plainText.equals(newString.toString())) {
```

```
        plainText = newString.toString();
        System.out.println("Modified plain text is ---- " + plainText);
    }
}

public int[] getDimensions(char letter) {
    int[] dimen = new int[2];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (key_mat[i][j] == letter) {
                dimen[0] = i;
                dimen[1] = j;
                break;
            }
        }
    }
    return dimen;
}

public void encryptMessage() {
    for (int i = 0; i < plainText.length(); i = i + 2) {
        int p1[] = new int[2];
        int p2[] = new int[2];
        p1 = getDimensions(plainText.charAt(i));
        p2 = getDimensions(plainText.charAt(i + 1));
        if (p1[0] == p2[0]) {
            int c1 = (p1[1] + 1) % 5;
            int c2 = (p2[1] + 1) % 5;
            cipherText = cipherText + key_mat[p1[0]][c1] + key_mat[p1[0]][c2];
        } else if (p1[1] == p2[1]) {
            int r1 = (p1[0] + 1) % 5;
            int r2 = (p2[0] + 1) % 5;
            cipherText = cipherText + key_mat[r1][p1[1]] + key_mat[r2][p1[1]];
        } else {
            cipherText = cipherText + key_mat[p1[0]][p2[1]] + key_mat[p2[0]][p1[1]];
        }
    }
}

public void decryptMessage() {
    plainText = "";
    for (int i = 0; i < cipherText.length(); i = i + 2) {
        int p1[] = new int[2];
        int p2[] = new int[2];
        p1 = getDimensions(cipherText.charAt(i));
        p2 = getDimensions(cipherText.charAt(i + 1));
```

```
            if (p1[0] == p2[0]) {
                int c1 = (p1[1] - 1 + 5) % 5;
                int c2 = (p2[1] - 1 + 5) % 5;
                plainText = plainText + key_mat[p1[0]][c1] + key_mat[p1[0]][c2];
            } else if (p1[1] == p2[1]) {
                int r1 = (p1[0] - 1 + 5) % 5;
                int r2 = (p2[0] - 1 + 5) % 5;
                plainText = plainText + key_mat[r1][p1[1]] + key_mat[r2][p1[1]];
            } else {
                plainText = plainText + key_mat[p1[0]][p2[1]] + key_mat[p2[0]][p1[1]];
            }
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        PlayFairCipher pfc = new PlayFairCipher();
        System.out.println("\nENCRYPTION");
        System.out.print("\nEnter plain text: ");
        pfc.plainText = sc.next();
        pfc.modifyPlainText();
        System.out.print("\nEnter the key: ");
        pfc.keyword = sc.next();
        pfc.validateKey();
        pfc.printKeyMatrix();
        pfc.encryptMessage();
        System.out.println("\nCipher text is: " + pfc.cipherText);
        System.out.println("\nDECRYPTION");
        System.out.print("\nEnter cipher text: ");
        pfc.cipherText = sc.next();
        System.out.print("\nEnter the key: ");
        pfc.keyword = sc.next();
        pfc.validateKey();
        pfc.printKeyMatrix();
        pfc.decryptMessage();
        System.out.println("\nPlain text is: " + pfc.plainText);
        sc.close();
    }
}
```

OUTPUT:

Example 1:

```
C:\Users\WELCOME\Desktop\CNS lab\ex1>java PlayFairCipher

ENCRYPTION

Enter plain text: instruments
Modified plain text is ---- instrumentsz

Enter the key: monarchy

The key matrix is

m o n a r
c h y b d
e f g i k
l p q s t
u v w x z

Cipher text is: gatlmzclrqtx

DECRYPTION

Enter cipher text: gatlmzclrqtx

Enter the key: monarchy

The key matrix is

m o n a r
c h y b d
e f g i k
l p q s t
u v w x z

Plain text is: instrumentsz

C:\Users\WELCOME\Desktop\CNS lab\ex1>
```

Example 2:

```
C:\Users\WELCOME\Desktop\CNS lab\ex1>java PlayFairCipher

ENCRYPTION

Enter plain text: jackandjill
After replacing j with i in the plain text ---- iackandiill
Modified plain text is ---- iackandiillz

Enter the key: jaihind
Modified key is ------ iahnd

The key matrix is

i a h n d
b c e f g
k l m o p
q r s t u
v w x y z

Cipher text is: ahblhdiaakpw

DECRYPTION

Enter cipher text: ahblhdiaakpw

Enter the key: iahnd

The key matrix is

i a h n d
b c e f g
k l m o p
q r s t u
v w x y z

Plain text is: iackandiillz

C:\Users\WELCOME\Desktop\CNS lab\ex1>
```

Example 3:

```
C:\Users\WELCOME\Desktop\CNS lab\ex1>java PlayFairCipher

ENCRYPTION

Enter plain text: balloon
Modified plain text is ---- balzloon

Enter the key: occurrence
Modified key is ------ ocuren

The key matrix is

o c u r e
n a b d f
g h i k l
m p q s t
v w x y z

Cipher text is: dbtegeng

DECRYPTION

Enter cipher text: dbtegeng

Enter the key: occurrence
Modified key is ------ ocuren

The key matrix is

o c u r e
n a b d f
g h i k l
m p q s t
v w x y z

Plain text is: balzloon

C:\Users\WELCOME\Desktop\CNS lab\ex1>
```