Objective 1

Write a program to implement the encryption and decryption process of the following Transposition techniques:

a. Rail Fence Cipher:

Procedure: Rail Fence Cipher is an transposition cipher which uses same key for both encryption and decryption. Process can be illustrated as follow.

Text: NITJALANDHAR

Key: 3

N_____A_____D

__I __ J___L____N___H____R

___T_____A _____A _

Code:

```
import java.util.Scanner;

public class RailFence {

	private static String encryption(String plain, int key) {
		String encryptedText = "";
		int col = plain.length();
		int row = key;
		boolean check = false;
		int j = 0;
		char[][] rail = new char[row][col];

		for (int i = 0; i < row; i++) {
			for (int k = 0; k < col; k++)
				rail[i][k] = '*';
		}

		for (int i = 0; i < col; i++) {
			if (j == 0 || j == key - 1)
				check = !check;
			rail[j][i] = plain.charAt(i);

			if (check)
				j++;
			else
				j--;
		}

		for (int i = 0; i < row; i++) {
			for (int k = 0; k < col; k++) {
```

```
                        char ch = rail[i][k];
                        if (ch != '*')
                                encryptedText += rail[i][k];
                }
        }
        return encryptedText;
}

private static String decryption(String encrypted, int key) {
        String decryptedText = "";

        int col = encrypted.length();
        int row = key;
        boolean check = false;
        int j = 0;
        char[][] rail = new char[row][col];

        // to fill the array:
        for (int i = 0; i < row; i++) {
                for (int k = 0; k < col; k++)
                        rail[i][k] = '*';
        }

        for (int i = 0; i < col; i++) {
                if (j == 0 || j == key - 1)
                        check = !check;
                rail[j][i] = '#';

                if (check)
                        j++;
                else
                        j--;
        }

        int index = 0;
        for (int i = 0; i < row; i++) {
                for (int k = 0; k < col; k++) {
                        char ch = rail[i][k];
                        if (ch == '#' && index < col) {
                                rail[i][k] = encrypted.charAt(index++);
                        }
                }
        }

        j = 0;
        check = false;
```

```
            for (int i = 0; i < col; i++) {
                if (j == 0 || j == key - 1)
                    check = !check;
                decryptedText += rail[j][i];

                if (check)
                    j++;
                else
                    j--;
            }

            return decryptedText;
        }

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the plain text message : ");
            String plainTxt = sc.nextLine();

            System.out.print("Enter the key : ");
            int key = sc.nextInt();

            String cipherTxt = encryption(plainTxt, key);
            System.out.println("\nEncrypted message: " + cipherTxt);

            String deCipherTxt = decryption(cipherTxt, key);
            System.out.println("\nDecrypted message: " + deCipherTxt);

        }
}
```

Output:

```
PS D:\DATAs\NITJ\CryptoLab> javac .\RailFence.java
PS D:\DATAs\NITJ\CryptoLab> java RailFence

Enter the plain text message : meet me at NIT Jalandhar
Enter the key : 4

Encrypted message: meIaem NTlnre a  adattJh

Decrypted message: meet me at NIT Jalandhar

PS D:\DATAs\NITJ\CryptoLab>
```

Objective 1

Write a program to implement the encryption and decryption process of the following Transposition techniques:

b. Keyless and keyed based combining approach

Code:

```
import java.io.*;
import java.util.*;

public class KeyedAndKeyless {

    static Scanner sc = new Scanner(System.in);
        public static String selectedKey;
        public static char sortedKey[];
        public static int sortedKeyPos[];

        public KeyedAndKeyless()
        {
      System.out.println(("Enter key for Keyed encryption: "));
                //selectedKey = "megabuck";
      selectedKey = sc.nextLine();
                sortedKeyPos = new int[selectedKey.length()];
                sortedKey = selectedKey.toCharArray();
        }

        public KeyedAndKeyless(String GeeksForGeeks)
        {
                selectedKey = GeeksForGeeks;
                sortedKeyPos = new int[selectedKey.length()];
                sortedKey = selectedKey.toCharArray();
        }

        public static void doProcessOnKey()
        {
                int min, i, j;
                char orginalKey[] = selectedKey.toCharArray();
                char temp;

                for (i = 0; i < selectedKey.length(); i++) {
                        min = i;
                        for (j = i; j < selectedKey.length(); j++) {
                                if (sortedKey[min] > sortedKey[j]) {
                                        min = j;
                                }
```

```
                }

                if (min != i) {
                        temp = sortedKey[i];
                        sortedKey[i] = sortedKey[min];
                        sortedKey[min] = temp;
                }
        }

        for (i = 0; i < selectedKey.length(); i++) {
                for (j = 0; j < selectedKey.length(); j++) {
                        if (orginalKey[i] == sortedKey[j])
                                sortedKeyPos[i] = j;
                }
        }
}

public static String doEncryption(String plainText)
{
        int min, i, j;
        char orginalKey[] = selectedKey.toCharArray();
        char temp;
        doProcessOnKey();

        int row = plainText.length() / selectedKey.length();
        int extrabit
                = plainText.length() % selectedKey.length();
        int exrow = (extrabit == 0) ? 0 : 1;
        int rowtemp = -1, coltemp = -1;
        int totallen = (row + exrow) * selectedKey.length();
        char pmat[][] = new char[(row + exrow)]
                                                [(selectedKey.length())];
        char encry[] = new char[totallen];

        int tempcnt = -1;
        row = 0;

        for (i = 0; i < totallen; i++) {
                coltemp++;
                if (i < plainText.length()) {
                        if (coltemp == (selectedKey.length())) {
                                row++;
                                coltemp = 0;
                        }
                        pmat[row][coltemp] = plainText.charAt(i);
                }
```

```
                else {

                        pmat[row][coltemp] = '-';
                }
        }
        int len = -1, k;

        for (i = 0; i < selectedKey.length(); i++) {
                for (k = 0; k < selectedKey.length(); k++) {
                        if (i == sortedKeyPos[k]) {
                                break;
                        }
                }
                for (j = 0; j <= row; j++) {
                        len++;
                        encry[len] = pmat[j][k];
                }
        }

        String p1 = new String(encry);
        return (new String(p1));
}

public static String doDecryption(String s)
{
        int min, i, j, k;
        char key[] = selectedKey.toCharArray();
        char encry[] = s.toCharArray();
        char temp;

        doProcessOnKey();

        int row = s.length();
        selectedKey.length();
        char pmat[][]
                = new char[row][(selectedKey.length())];
        int tempcnt = -1;

        for (i = 0; i < selectedKey.length(); i++) {
                for (k = 0; k < selectedKey.length(); k++) {
                        if (i == sortedKeyPos[k]) {
                                break;
                        }
                }
```

```
                for (j = 0; j < row; j++) {
                        tempcnt++;
                        pmat[j][k] = encry[tempcnt];
                }
        }

        char p1[] = new char[row * selectedKey.length()];

        k = 0;
        for (i = 0; i < row; i++) {
                for (j = 0; j < selectedKey.length(); j++) {
                        if (pmat[i][j] != '*') {
                                p1[k++] = pmat[i][j];
                        }
                }
        }
        p1[k++] = '\0';
        return (new String(p1));
    }

    @SuppressWarnings("static-access")

    public static void main(String[] args)
    {
  KeyedAndKeyless kCipher = new KeyedAndKeyless();

        System.out.print("\nEnter the plain text message : ");
        String plainTxt = sc.nextLine();

        System.out.println("Cipher Text : "
                                + kCipher.doEncryption(plainTxt));
    }
}
```

Output:

```
PS D:\DATAs\NITJ\CryptoLab> javac .\KeyedAndKeyless.java
PS D:\DATAs\NITJ\CryptoLab> java KeyedAndKeyless
Enter key for Keyed encryption:
qwerty

Enter the plain text message : Meet me at library
Cipher Text : earMeitta  re bmly
PS D:\DATAs\NITJ\CryptoLab>
```