Ex. No. 3	**Column Transposition and Rail Fence Cipher**
Date of Exercise	17.08.2023

1) Aim

To implement the Rail Fence Cipher.

Description:

The Rail Fence Cipher, also known as the Zigzag Cipher, is a simple transposition cipher that was historically used for basic message encryption. The fundamental concept behind this encryption method involves writing the plaintext message in a zigzag pattern along a set number of "rails" or rows. Each rail represents a row in the pattern. Once the message is written out in this zigzag fashion, it's read from left to right, row by row, to create the ciphertext. This rearrangement of characters makes it an example of a classical columnar transposition cipher.

To decrypt a message encrypted using the Rail Fence Cipher, the recipient needs to know the number of rails or rows used for encryption. Then, the recipient reconstructs the zigzag pattern and reads the characters following the same zigzag path to reveal the original plaintext.

The Rail Fence Cipher is relatively easy to understand and implement, making it a valuable educational tool for introducing basic encryption concepts. However, it's considered a weak encryption method and is not suitable for secure communication in modern contexts due to its susceptibility to decryption through various cryptanalysis techniques. More robust and complex encryption algorithms have since replaced it in practical cryptography.

Algorithm:

Step 1: Input: Prompt the user to enter a string s to be encrypted and an integer k representing the key for the Rail Fence Cipher.

Step 2: Initialize a 2D list enc with dimensions k rows and len(s) columns, filled with empty spaces.

Step 3: Initialize a flag flag to 0, and a variable row to 0. These will be used to control the zigzag pattern.

Step 4: Loop through the characters of the input string s:

a. Place the current character of s in the enc array at the current row and column.

b. Update the row and flag to control the zigzag pattern. If row is 0, set flag to 0; if row is k-1, set flag to 1.

c. If flag is 0, increment row by 1; if flag is 1, decrement row by 1.

Step 5: Print the Rail Fence Cipher matrix enc row by row.

Step 6: Initialize an empty list ct to store the cipher text.

Step 7: Loop through the rows of the enc array and the columns of the input string s:

a. If the character in the enc array is not an empty space, append it to the ct list.

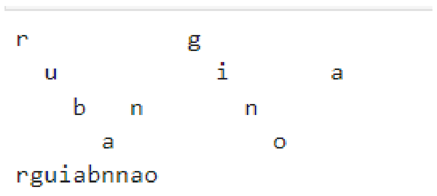
Step 8: Join the characters in the ct list to form the cipher text.

Step 9: Print the resulting cipher text.

Program:

```
s = input("Enter the String: ")

k = int(input("Enter the key: "))


enc = [[" " for i in range(len(s))] for j in range(k)]


flag = 0

row = 0


for i in range(len(s)):

    enc[row][i] = s[i]


    if row == 0:

        flag = 0

    elif row == k-1:

        flag = 1


    if flag == 0:

        row += 1

    else:
        row -= 1


for i in range(k):
    print(" ".join(enc[i]))


ct = []
for i in range(k):
    for j in range(len(s)):
        if enc[i][j] != ' ':
            ct.append(enc[i][j])
print("".join(ct))
```

Output Screenshot:



```
r           g
  u           i       a
    b   n       n
      a           o
rguiabnnao
```

2) Aim:

To implement a Column Transposition Cipher.

Description:

In cryptography, a transposition cipher (also known as a permutation cipher) is a method of encryption which scrambles the positions of characters (transposition) without changing the characters themselves. Transposition ciphers reorder units of plaintext (typically characters or groups of characters) according to a regular system to produce a ciphertext which is a permutation of the plaintext. They differ from substitution ciphers, which do not change the position of units of plaintext but instead change the units themselves. Despite the difference between transposition and substitution operations, they are often combined, as in historical ciphers like the ADFGVX cipher or complex high-quality encryption methods like the modern Advanced Encryption Standard (AES).

Algorithm:

Step 1: Input: Prompt the user to enter a message and a key for encryption.

Step 2: Initialize an empty string called cipher to store the encrypted message.

Step 3: Calculate the length of the message and store it in the variable msg_len.

Step 4: Convert the message string into a list called msg_lst.

Step 5: Sort the characters of the key and store them in a list called key_lst.

Step 6: Determine the number of columns needed for the transposition by taking the length of the key.

Step 7: Calculate the number of rows required for the transposition as the ceiling value of the division of the message length by the number of columns.

Step 8: Calculate the number of empty spaces needed to fill the last row of the transposition matrix. This is done by subtracting the message length from the product of the number of rows and columns. Extend the msg_lst by appending underscores (_) to fill these empty spaces.

Step 9: Create a matrix (2D list) called matrix by splitting the extended msg_lst into rows, with each row having a length equal to the number of columns.

Step 10: Initialize a variable key_index to 0.

Step 11: For each column in the transposition matrix, do the following:

a. Find the index of the character from the sorted key list in the original key list, and store it in current_index.

b. Append the characters from the transposition matrix at the current_index column to the cipher string.

c. Increment key_index to move on to the next character in the key.

Step 12: Return the cipher as the encrypted message.

Step 13: Print the "Encrypted Message" followed by the value of the cipher.

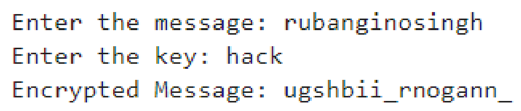
Program:

```
import math



def encryptMessage(message):
```

```
cipher = ""

key_index = 0


msg_len = float(len(message))

msg_lst = list(message)

key_lst = sorted(list(key))


column = len(key)


row = int(math.ceil(msg_len / column))


fill_null = int((row * column) - msg_len)

msg_lst.extend('_' * fill_null)


matrix = [msg_lst[i: i + column] for i in range(0, len(msg_lst), column)]


for _ in range(column):

    current_index = key.index(key_lst[key_index])

    cipher += ''.join([row[current_index]

                      for row in matrix])
```

```
        key_index += 1


    return cipher


message = input("Enter the message: ")

key=input("Enter the key: ")


cipher = encryptMessage(message)

print("Encrypted Message: {}". format(cipher))
```

Output Screenshot:



```
Enter the message: rubanginosingh
Enter the key: hack
Encrypted Message: ugshbii_rnogann_
```

Result

Thus, the experiment to Implement Rail Fence Cipher and Column Transposition Cipher is carried out successfully and obtained the required output.