

Ex. No. 3	Implement Transposition Cipher
Date of Exercise	29.08.2024

Aim

To implement the rail fence transposition technique using Python language.

Description

In the rail fence cipher, the plain text is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows.

EXAMPLE:

```
AUTHOR
1 6 5 2 3 4
-----
WEARED
ISCOVE
REDSAV
EYOURS
ELFABC
```

yields the cipher

WIREEROSUA EVARBDEVSCACDOFESEYL.

1. Perform encryption and decryption using rail fence cipher**Algorithm**

STEP-1: Read the Plain text.

STEP-2: Arrange the plain text in row columnar matrix format.

STEP-3: Now read the keyword depending on the number of columns of the plain text.

STEP-4: Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row-wise or column-wise in the former order to get the cipher text.

Program

```
print("URK21CS1128")

def rail_fence_cipher(text,key):

    rails = [[] for _ in range(key)]

    rail = 0

    direction = 1

    for char in text:

        rails[rail].append(char)

        rail += direction

        if rail == 0 or rail == key-1:

            direction *= -1

    return ".join(['.join(rail) for rail in rails])

def rail_fence_decipher(cipher_text,key):

    rails = [[] for _ in range(key)]

    rail = 0
```

```
direction = 1

for _ in cipher_text:

    rails[rail].append('*')

    rail += direction

    if rail == 0 or rail == key-1:

        direction *= -1

index = 0

for i in range(key):

    for j in range(len(rails[i])):

        rails[i][j] = cipher_text[index]

        index += 1

plain_text = []

rail = 0

direction = 1

for _ in cipher_text:

    plain_text.append(rails[rail].pop(0))

    rail += direction

    if rail == 0 or rail == key - 1:

        direction *= -1

return ".join(plain_text)
```

```
input_string = input("Enter the string: ")
```

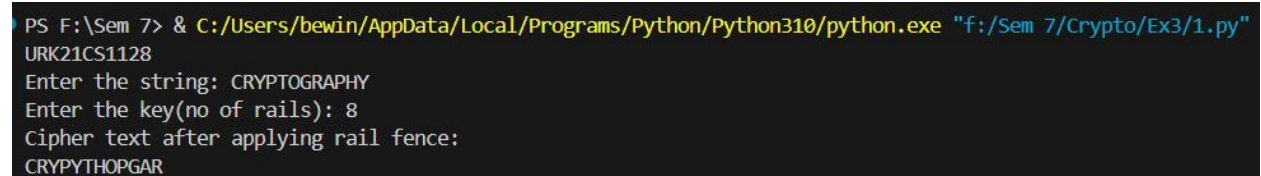
```
key = int(input("Enter the key(no of rails): "))
```

```
cipher_text = rail_fence_cipher(input_string,key)

print("Cipher text after applying rail fence: ")

print(cipher_text)
```

Output Screenshot



```
PS F:\Sem 7> & C:/Users/bewin/AppData/Local/Programs/Python/Python310/python.exe "f:/Sem 7/Crypto/Ex3/1.py"
URK21CS1128
Enter the string: CRYPTOGRAPHY
Enter the key(no of rails): 8
Cipher text after applying rail fence:
CRYPYTHOPGAR
```

2. Perform encryption using the columnar transposition technique

Algorithm

STEP 1: Read the problem statement and take the input as plain text and key.

STEP 2: Generate a list of key characters with their indices and sort them.

STEP 3: Create and fill the matrix with equal to the length of the plain text divided by the key length.

STEP 4: Read the matrix column by column based on the sorted key order.

STEP 5: Concatenate the characters from the columns to generate the final encrypted text.

Program:

```
print("URK21CS1128")

pt = input("Plain text: ").replace(" ", "")

key = sorted(list(enumerate(list(input("key: ")))),key=lambda x:ord(x[1]))

m = []

row = 0

index = 0

while index != len(pt):
```

```
m.append([""]*len(key))

for col in range(len(key)):

    m[row][col] = pt[index]

    index += 1

    if index == len(pt):

        break

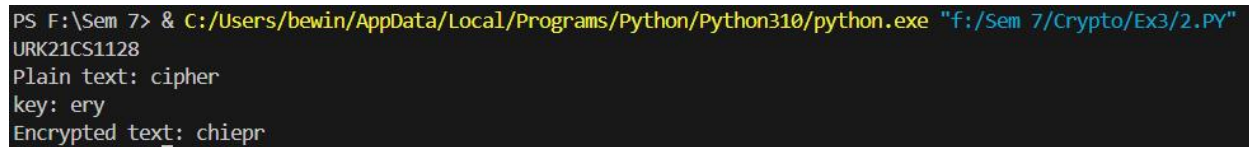
    row+=1

print("Encrypted text:",end=" ")

for col,value in key:

    for row in range(len(m)):

        print(m[row][col],end="")
```

Output:

```
PS F:\Sem 7> & C:/Users/bewin/AppData/Local/Programs/Python/Python310/python.exe "f:/Sem 7/Crypto/Ex3/2.PY"
URK21CS1128
Plain text: cipher
key: ery
Encrypted text: chiepr
```

Result

The program has executed successfully and the output is displayed in the console.