Practical 5

Implement rail Fence and transposition cipher. Both are permutation cipher. Analyze the strength of the cipher in terms of cryptanalysis.

- **Rail fence.**

Implement rail Fence cipher.

M		e		m	
	e		t		e

Test case : Meetme
Ciphertext : MEMETE

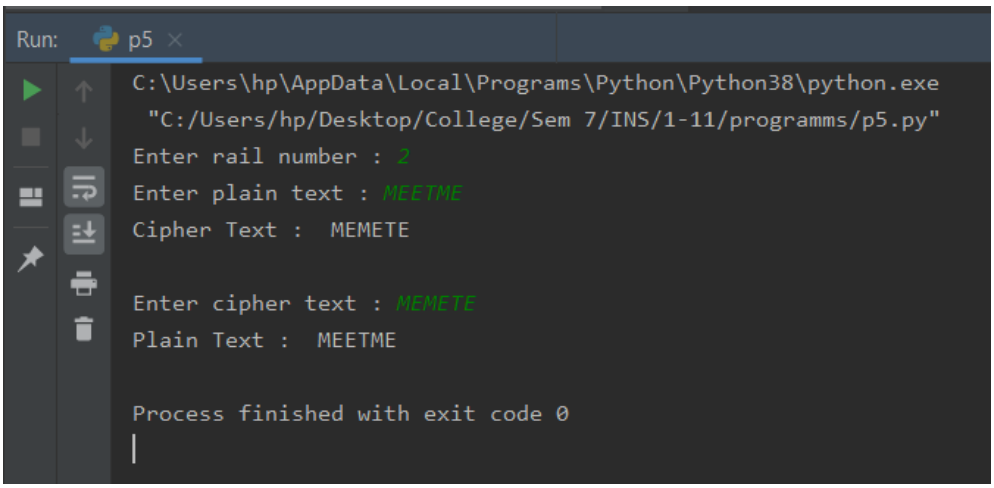
Given the ciphertext retrieve the plaintext.
Test Case: MEMETE
Plaintext: Meetme

Program:

```
def encrypt(plain,rail):
    cipher = ""
    out = {}
    cycle = (2*rail) - 2
    for i in range(rail):
        out[i]=""
        index = i
        first = cycle - i*2
        second = cycle - first
        while index < len(plain):
            if first !=0:
                out[i] += plain[index]
            index += first
            first, second = second, first

    for i in range(rail):
        cipher += out[i]
    return cipher

def decrypt(cipher,rail):
    plain = [None]*len(cipher)
    cycle = (2*rail) - 2
```

```
    point = 0
    for i in range(rail):

        index = i
        first = cycle - i*2
        second = cycle - first

        while index < len(cipher):
            if first !=0:
                plain[index] = cipher[point]
                point += 1
            index += first

            first, second = second, first
    return "".join(plain)

rail = int(input("Enter rail number : "))
plain = input("Enter plain text : ")

c = encrypt(plain,rail)
print("Cipher Text : ",c)
c = input("\nEnter cipher text : ")
print("Plain Text : ",decrypt(c,rail))
```

Output:



- **Transposition cipher**

Implement Transposition cipher

Key : 4312567
Plaintext: attackpostponeduntiltwoam
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Given the ciphertext, get the plaintext back.

Program:

```
import math
a = "abcdefghijklmnopqrstuvwxyz"

def encrypt(k):
    plain = input("\nEnter plain text : ")
    cols = len(k)
    rows = math.ceil(len(plain)/cols)
    d = {}
    for i in k:
        d[int(i)]=""

    random_input = rows*cols - len(plain)

    for i,c in enumerate(plain):
        d[int(k[i%cols])] += c

    for i in reversed(range(1,random_input+1)):
        d[int(k[cols-i])] += a[-i]

    print("Cipher Text - ",end="")

    for i in range(1,cols+1):
        print(d[i].upper(),end="")
    print()

def decrypt(k):
```

```
    cipher = input("\nEnter cipher text : ")
    cols = len(k)
    row = math.ceil(len(cipher)/cols)
    rows = []
    ans = []
    for i in range(len(cipher)):
        if i%row==0:
            rows.append(list(cipher[i:i+row]))
    for i in range(row):
        for j in k:
            ans.append(rows[int(j)-1].pop(0))
    print("Plain Text : ", "".join(ans).lower())

k = input("Enter key : ")
encrypt(k)
decrypt(k)
```

Output: