Exp: 1D **Columnar Transposition Techniques**
Date: 24-02-2024

Aim:

To write a python program implementing columnar transposition techniques.

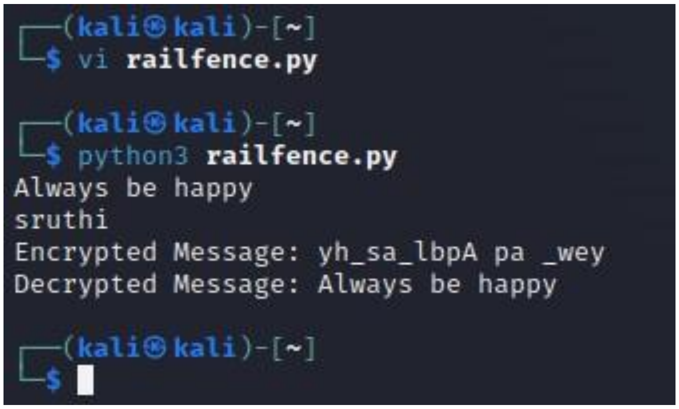
Algorithm:

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. The permutation is defined by the alphabetical order of the letters in the keyword.
4. Any spare spaces are filled with nulls or left blank or placed by a character (Example: _).
5. Finally, the message is printed off in columns, in the order specified by the keyword. **Program:** import math def encryptMessage(msg,key):

```
    cipher = "" k_indx = 0 msg_len =
    float(len(msg)) msg_lst = list(msg) key_lst =
    sorted(list(key)) col = len(key) row =
    int(math.ceil(msg_len / col)) fill_null = int((row
    * col) - msg_len) msg_lst.extend('_' * fill_null)
    matrix = [msg_lst[i: i + col] for i in range(0,
    len(msg_lst), col)]
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx]
                                                for row in matrix])
        k_indx += 1
    return cipher
def decryptMessage(cipher,key):
    msg = "" k_indx = 0
    msg_indx = 0
    msg_len =
    float(len(cipher)) msg_lst =
    list(cipher) col = len(key)
    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))
    dec_cipher = [] for _ in
    range(row): dec_cipher +=
    [[None] * col]
    for _ in range(col): curr_idx = key.index(key_lst[k_indx])
        for j in range(row): dec_cipher[j][curr_idx] =
        msg_lst[msg_indx] msg_indx += 1
        k_indx += 1
    try:
        msg = ".join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                                "handle repeating words.")
    null_count = msg.count('_') if
    null_count > 0: return msg[: -
    null_count]
    return msg

msg = input() key=input() cipher =
encryptMessage(msg,key)
print("Encrypted Message: {}".
format(cipher))
print("Decrypted Message: {}".
    format(decryptMessage(cipher,key)))
```

Output:



```
┌──(kali㉿kali)-[~]
└─$ vi railfence.py

┌──(kali㉿kali)-[~]
└─$ python3 railfence.py
Always be happy
sruthi
Encrypted Message: yh_sa_lbpA pa _wey
Decrypted Message: Always be happy

┌──(kali㉿kali)-[~]
└─$ 
```

Result:

Thus the python program for columnar transposition techniques is implemented successfully.