In [3]:

```
#Simran Sayyad
#COTB52
#lab 3
#railfence cipher
def encryptRailFence(text, key):
    rail = [['\n' for i in range(len(text))]
                for j in range(key)]

    dir_down = False
    row, col = 0, 0

    for i in range(len(text)):

        if (row == 0) or (row == key - 1):
            dir_down = not dir_down


        rail[row][col] = text[i]
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    result = []
    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])
    return("" . join(result))

def decryptRailFence(cipher, key):

    rail = [['\n' for i in range(len(cipher))]
                for j in range(key)]

    dir_down = None
    row, col = 0, 0

    for i in range(len(cipher)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False

        rail[row][col] = '*'
        col += 1

        if dir_down:
            row += 1
        else:
            row -= 1

    index = 0
    for i in range(key):
        for j in range(len(cipher)):
            if ((rail[i][j] == '*') and
            (index < len(cipher))):
                rail[i][j] = cipher[index]
                index += 1
    result = []
    row, col = 0, 0
    for i in range(len(cipher)):

        if row == 0:
            dir_down = True
        if row == key-1:
            dir_down = False

        if (rail[row][col] != '*'):
            result.append(rail[row][col])
            col += 1


        if dir_down:
            row += 1
        else:
            row -= 1
    return("".join(result))


if __name__ == "__main__":
    print(encryptRailFence("attack at once", 2))
    print(encryptRailFence("information security ", 3))
    print(encryptRailFence("defend the east wall", 3))


    print(decryptRailFence("atc toctaka ne", 2))
    print(decryptRailFence("dnhaweedtees alf tl", 3))
```

```
atc toctaka ne
irisr nomto euiyfanct
dnhaweedtees alf  tl
attack at once
delendfthe east wal
```

In [4]:

```
# Columnar Transposition
import math

key = "HACK"

# Encryption
def encryptMessage(msg):
    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

# Decryption
def decryptMessage(cipher):
    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 = "Information security"

cipher = encryptMessage(msg)
print("Encrypted Message: {}".
            format(cipher))

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

```
Encrypted Message: nmoeifanctIrisrot uy
Decryped Message: Information security
```

```
In [ ]:
```