Implementation Columnar Transposition Cipher

The columnar transposition cipher is a fairly simple, easy to implement cipher. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the ciphertext.

Although weak on its own, it can be combined with other ciphers, such as a substitution cipher, the combination of which can be more difficult to break than either cipher on it's own.

Importing required packages

In [3]:

```
1 import math
```

Taking input from user

In [12]:

```
1 key=input("Enter keyword text (Contains unique letters only): ").lower().replace(" ", "
2 plain_text = input("Enter plain text (Letters only): ").lower().replace(" ", "")
3
4 len_key = len(key)
5 len_plain = len(plain_text)
6 row = int(math.ceil(len_plain / len_key))
7 matrix = [ ['X']*len_key for i in range(row) ]
8
9 print(matrix)
```

```
Enter keyword text (Contains unique letters only): heo
Enter plain text (Letters only): tarun
[['X', 'X', 'X'], ['X', 'X', 'X']]
```

Encryption

Our message exactly fit the rectangular array. If the message does not completely fill the array, nulls (i.e., padding) may be added to fill the array (this is the easier cipher to break) or not (this is harder to break because the columns do not all have the same length). In the latter case, the length of the keyword determines the number of columns, and the number of letters in the message determines the number of complete and partial rows

In [13]:

```
t = 0
for r in range(row):
  for c,ch in enumerate(plain_text[t : t+ len_key]):
    matrix[r][c] = ch
  t += len_key

# print(matrix)
sort_order = sorted([(ch,i) for i,ch in enumerate(key)])  #to make alphabetically order
# print(sort_order)

cipher_text = ''
for ch,c in sort_order:
  for r in range(row):
    cipher_text += matrix[r][c]

print("Encryption")
print("Plain text is :",plain_text)
print("Cipher text is:",cipher_text)
```

```
Encryption
Plain text is : tarun
Cipher text is: anturX
```

Decryption

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then, write the message out in columns again, then re-order the columns by reforming the key word.

In [14]:

```
matrix_new = [ ['X']*len_key for i in range(row) ]
key_order = [ key.index(ch) for ch in sorted(list(key))]  #to make original key order w
# print(key_order)

t = 0
for c in key_order:
  for r,ch in enumerate(cipher_text[t : t+ row]):
    matrix_new[r][c] = ch
  t += row
# print(matrix_new)

p_text = ''
for r in range(row):
  for c in range(len_key):
    p_text += matrix_new[r][c] if matrix_new[r][c] != 'X' else ''

print("Decryption")
print("Cipher text is:",cipher_text)
print("Plain text is :",p_text)
```

```
Decryption
Cipher text is: anturX
Plain text is : tarun
```