In [ ]:

```
import re

def text_to_numerical_val_list(text):
    regex = re.compile('[^a-zA-Z]') # First we would like to remove all non-alphabetical characters such as commas, periods, exclamation marks, etc.
    text = regex.sub('', text)
    text = re.sub(r"\s+", "", text, flags=re.UNICODE)# We also want to remove any spacing
    text = text.lower() # important since there is a distinction beetwen uppercase and lowercase alphabet characters
    text_char_to_numerical_value_list = []
    for character in text:
        numerical_value = ord(character) - 97 # we take the numeric value then subract 97 to fit the numbers within the regular alphabet
        text_char_to_numerical_value_list.append(numerical_value)
    return text_char_to_numerical_value_list
```

The code above takes in a string wich is either plain text or cipher text and coverts that string into an array of values based on the numeric index of that character in the alphabet - 1. Ex. the function takes in the following string, text_to_numerical_list("The string can consist of plaintext or CIPHERTEXT") and the returned value will be an array of integer values, as follows [19, 7, 4, 18, ..., n]. The resulting array has n elements, n is essentailly the length of the string with all non-alphabetical characters and spaces removed, notice T is the 20th letter in the alphabet, but corresponds to 19 in the array since we start indexing from 0.

In [ ]:

```
def encipher(plaintext,keys = (13,11)): # takes in a string containing the plaintext and an ordered pair of alpha and beta
    plaintext_numerical_val_list= text_to_numerical_val_list(plaintext)
    alpha = keys[0]
    beta = keys[1]
    cipher_values = []
    for i in plaintext_numerical_val_list:
        cipher_val = (alpha*i)+beta # the alpha value is multiplied to the numeric index of a given charater in the given string
        cipher_values.append(chr(((cipher_val)% 26)+97))
    cipher_text = ''.join(str(char) for char in cipher_values)
    return "Cipher text:  "+cipher_text.upper()
```

The code above is used to encipher a given plaintext using the provided keys. The keys consist of an α and a β where α must be co-prime with 26, that is gcd(α,26) = 1, and $0 < \alpha, \beta < 26$. In terms of a function for the code above: $f(p) = (\alpha p) + \beta, p$ is the numeric value of a letter, then the corresponding cipher value $C = f(p) \mod 26$.

In [ ]:

```
def decipher(ciphertext,keys):
    cipher_text_to_numerical_val_list = text_to_numerical_val_list(ciphertext)
    print(cipher_text_to_numerical_val_list)
    alpha = keys[0]
    beta = keys[1]
    alpha_inverse = pow(alpha, -1, 26)#built in extended euclidean algorithm for finding the modular inverse
    deciphered_values = []
    for i in cipher_text_to_numerical_val_list:
        deciphered_val = alpha_inverse*(i-beta) # the modular inverse of alpha is multiplied to the cipher value - beta
        deciphered_values.append(chr(((deciphered_val)% 26)+97))
    deciphered_text = ''.join(str(char) for char in deciphered_values)
    return "Deciphered text:  "+deciphered_text.lower()
```

The code above is used to decipher a given plaintext using the provided keys. Again the keys consist of an α and a β where α must be co-prime with 26, and $0 < \alpha, \beta < 26$. As a function: $f(C) = \alpha^{-1}(C - \beta)$, where α^{-1} is the modular inverse of alpha which is equal to κ, where $\alpha \cdot \kappa \equiv 1 (\mod 26)$, this κ is derived using the extended euclidean algorithm, the κ value is multiplied to $C - \beta$.

Obtaining the deciphered plaintext value, $p = f(C) \mod 26$

In [ ]:

```
print(encipher("Affine's cipher method is not secure nor more secure than Vigener's cipher and should only be used for fun",(3,11))) # takes in a paramter of strings for plaintext
#Terminal output - Cipher text: LAAJYXNRJEGXKVXQGBUJNYBQNXRTKXYBKVBKXNXRTKXQGLYWJDXYXKNRJEGXKLYUNGBTSUBYSFOXTNXUABKAT
print(decipher("PJXFJSWJNXJMRTJFVSUJOOJWFOVAJRWHEOFJRWJODJFFZBJF",(9,25))) # takes in a paramter of strings and ordered pair for key
#Terminal output - Deciphered text:  we use frequencies of letters to decrypt secret messages
```

```
Cipher text:  LAAJYXNRJEGXKVXQGBUJNYBQNXRTKXYBKVBKXNXRTKXQGLYWJDXYXKNRJEGXKLYUNGBTSUBYSFOXTNXUABKATY
[15, 9, 23, 5, 9, 18, 22, 9, 13, 23, 9, 12, 17, 19, 9, 5, 21, 18, 20, 9, 14, 14, 9, 22, 5, 14, 21, 0, 9, 17, 22, 7, 4, 14, 5, 9, 17, 22, 9, 14, 3, 9, 5, 5, 25, 1, 9, 5]
Deciphered text:  weusefrequenciesofletterstodecryptsecretmessages
```