# An extended affine cipher

## Problem description

In this problem, we consider a modified affine cipher where the encryption of a letter in an input text is dependent on the position of the occurrence of that letter in the text. The plaintext used in this cipher are drawn from the following symbols: numerical digits (0-9), lowercase letters (a-z), upper case letters (A-Z) and the following symbols: '<' (less-than), '>' (greater-than), '[' (left bracket), ']' (right bracket) and '=' (equal sign). So there are a total of 67 characters that can be used in the plaintext. These characters are mapped to integers based on their positions in the following string:

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ<>[]=
```

So for example, the character '0' is mapped to 0, '1' to 1, 'a' to 10, 'A' to 36, '<' to 62, and so on.

Recall that in an affine cipher, the key consists of two components, which we refer to as a and b. In the modified affine cipher for this assignment, in encrypting a plaintext, we encrypt individual letters according to their position in the text (counting the position starting from 0). Given the key k=(a,b), the encryption function for a letter x at position i:

$$e_k(x, i) = a^i x + bi \mod 67.$$

Similarly, the decryption function for a letter y at position i is

$$d_k(y, i) = (y - bi)a^{-i} \mod 67.$$

For example, if the key is (12,3) and the plaintext is `thisisatest`, then the encrypted text is `t6Qj3MznZI2`. Note that the first letter is never encrypted (since for $i = 0$, $a^i x + bi \mod 67 = x$).

Your task is to decrypt the file cipher.txt and obtain the flag, which is the first 10 characters in the plaintext. The flag is a concatenation of two English words, e.g., `easyflag`. The rest of the plaintext was a randomly generated string.

You must use an analytical method to solve this challenge, without using brute force search on the key space.

To help you solve this problem, a 'hint' file (hint.txt) is given, where the plaintext characters in some positions are revealed. In the hint file, the character '_' (underscore) denotes an unrevealed character. So using the example plaintext and ciphertext above, a hint file could contain something like:

```
__i_is_____
```

**HINT: Use Fermat's Little Theorem to simplify the equations you derive from the provided plaintext-ciphertext pairs.**

# Implementation of the extended affine cipher

In the file `affine.py` you will find an implementation of a position-dependent affine cipher described in the previous section. You can use this implementation to help you solve this challenge.

An encryption key in this cipher is a pair of numbers (a,b), where 0 < a,b < 67.

## System requirements

You need to have Python 3 and the Python library Pycryptodome installed. Pycryptodome can be installed using pip3:

```
pip3 install pycryptodome
```

## Encrypting a file

To encrypt a file, say, plain.txt, with the key k=(12,27), run:

```
python3 affine.py enc 12 27 plain.txt cipher.txt
```

This will produce the ciphertext and save it to the file cipher.txt.

## Decrypting a file

To decrypt a file, say, cipher.txt, with the key k=(12,27), run:

```
python3 affine.py dec 12 27 cipher.txt decrypted.txt
```

This will produce the plaintext and save it to the file decrypted.txt.