# Cracking The Caesar Cipher – Description and Requirements

### **Cracking the Caesar cipher**

Suppose we are given a cipher text, i.e. text that has already been encrypted with some (unknown) shift, and we want to determine the original unencrypted text (typically referred to as the plaintext). To reconstruct the original text we could decode with each of the 26 possible shifts, and take the result that looks “closest” to an English sentence.

How do we measure closeness? This is where letter frequencies and a small statistical trick comes in. First of all, we know how often each letter occurs on average in English text. For instance, ‘e’ is the most common letter, then ‘t’, and so on. To decode our cipher text, we can compute the frequencies of each letter as they appear in the text. To measure how close these are to the known English letter frequencies, we use the X2-score (that is the Greek letter “chi”, so it’s the “chi-squared score”). We sum the fraction over all 26 possible letters to determine this score, which is a single number.

The X2 score will be lower when the frequencies are closer to English. Note that we are ignoring the case of letters (we treat upper and lower case as equal) when computing the X2 score.

You are [provided with a file called](https://canvas.liverpool.ac.uk/courses/59716/files/8205026?wrap=1" \o "Brutus.java" \t "_blank)Brutus.java that already defines letter frequencies in English texts as an array of doubles, in alphabetical order:

public static final double[] english = {

0.0855, 0.0160, 0.0316, 0.0387, 0.1210, 0.0218, 0.0209, 0.0496, 0.0733,

0.0022, 0.0081, 0.0421, 0.0253, 0.0717, 0.0747, 0.0207, 0.0010, 0.0633,

0.0673, 0.0894, 0.0268, 0.0106, 0.0183, 0.0019, 0.0172, 0.0011

};

Accordingly, the frequency of the letter ‘a’ is *Englisha = 0.0855* and can be accessed as english[0]. Similarly, *Englishb = 0.0160* and so on.

#### **Requirements**

In Brutus.java, write (public static) methods as follows.

* a method called count that takes a single String parameter and returns a length-26 integer array (int[]) whose values reflect how often each character occurred. You should not make a difference between upper and lower case letters and the returned array should be in alphabetical order. This way, if letterCounts is an array resulting from your method then letterCounts[25] is the number of times the letter ‘z’ or ‘Z’ occurs.
* a method called frequency that takes a single String and returns a length-26 array of doubles whose values correspond, in alphabetical order, to the frequency of the letter. This way, if letterFreq is an array resulting from this method then letterFreq[24] is the number of times the letter ‘y’ or ‘Y’ occurs, divided by the size of the string input.[5]
* a method called chiSquared, which returns the X2-score (a double) for two given sets of frequencies. That is, it should take two parameters, both of type double[], and return a single double value that tells us how close these two arrays are. You may assume that the two inputs are always of length 26 and that the second input contains no zeros (we will use the given array for English letter frequencies in tests here).
* a main method that can be used to decipher Caesar-encoded English cryptotext without the key. Of course, you should be using your chiSquared method as well as the given English letter frequencies.

The string that is to be deciphered should be read from the first command line argument and your program should ensure that it gets exactly this one argument and complain otherwise. Sample output below.

$> java Brutus "Vg vf n zvfgnxr gb guvax lbh pna fbyir nal znwbe ceboyrzf whfg jvgu cbgngbrf."

It is a mistake to think you can solve any major problems just with potatoes.

$> java Brutus

Too few parameters!

Usage: java Brutus "cipher text"

$> java Brutus Too Many Parameters

Too many parameters!

Usage: java Brutus "cipher text"