4.1 Modify the function `search` from the course Python slides that searches for text in a file to *count the number of lines* in which the text appears. Example (from the theoretical class): `search("Love", "sonnets.txt")` returns 18.

4.2 Write a program that reads the file of *Sonnets* (available on the course's webpage) and searches for the length of the longest word.
Hint: use the `split()` method from strings to split a line into a list of words.

4.3 Write a program that reads the file of *Sonnets* (available on the course's webpage) and determines the *average length* of the words in the text: the sum of the lengths of each word divided by the number of words.
Hint: use the `split()` method from strings to split a line into a list of words.

4.4 Using the `collatz(n)` function (from the exercise of the previous lab class), define a program that writes a text file `"table.txt"` with a table of values for n and the *length* of `collatz(n)` from 1 to 1000; the first lines are:

```
n       len(collatz(n))
1       1
2       2
3       8
4       3
5       6
6       9
```

Use Python matplotlib to plot the graph of these points.

4.5 Check the response of the *Python* interpreter (in an interactive session) to each of the following lines:

```
>>> d = {'apples': 15, 'bananas': 35, 'grapes': 12}
>>> d['bananas']
>>> d['oranges'] = 20
>>> len(d)
>>> 'grapes' in d
>>> d['pears']
>>> d.get('pears', 0)
>>> sorted(d)
>>> del d['apples']
>>> 'apples' in d
```

Make sure you understand each of the results.

4.6 Write a function `conta_letras(txt)` that prints a table with the number of occurrences of each letter in the string `txt`, in alphabetical order. Uppercase and lowercase letters should be considered equivalent. Example:

```
>>> conta_letras("Uma luz do sol muito amarela")
a : 4
d : 1
e : 1
...
```

4.7 Write a function `maisFreq(txt)` that, given a string `txt`, returns the character that occurs most frequently in `txt`, written in *uppercase*. For this count, there should be no distinction between uppercase and lowercase letters. If there is more than one character with the highest occurrence count in `txt`, the function should return the character with the *smallest* lexicographic order.

Examples:

```
>>> maisFreq('excecionalmente')
'E'
>>> maisFreq('Inconstitucional')
'I'
```

4.8 Two words or phrases are *anagrams* if they are composed of the same letters used the same number of times, but possibly in different positions. For example, the Latin phrase "Quid est veritas?" (*What is truth?*) is an anagram of "Est vir qui adest" (*It is the man who is present*).

Write a function `anagrams(txt1,txt2)` that verifies if two strings are anagrams; the result should be `True` or `False`. It should consider upper and lowercase letters as equivalent, and ignore all non-letter characters (spaces, punctuation marks, etc.); you can also assume that the strings do not contain accented characters.