TPOP PRACTICAL

FUNCTIONS, LISTS AND DICTIONARIES

PRACTICAL 04

Question 1: *reinventing the wheel!*

For this question we are emulating the method `split()` from the type `str`. This exercise is more challenging than it may look like. It is crucial that you devise an algorithm before you start implementing. Simulate your algorithm on paper before writing any code. There are methods in the `str` type that can help you identify between an alphabet character and a punctuation, this might be very useful. If you are stuck, ask myself or one of the PTA to have look at your algorithm.

Using the file `practical_4_part1.py`, write and test a function to meet the following specification.

1. `splitText(text)` where `text` is a string and return the list of the words by splitting the string `text`. See example below:

```
>>> sampleText = "As Python's creator, I'd like to say a
few words about its origins.”
>>> splitText(sampleText)
['As', 'Python', 's', 'creator', 'I', 'd', 'like', 'to',
'say', 'a', 'few', 'words', 'about', 'its', 'origins']
```

You must NOT use the method `split()` from the `str` type, however other methods from the class are allowed. You must not use python library such as `string.py`.

Question 2: *from a resit paper.*

You must use the file `practical_4_part1.py` to answer this question. Write a function `getWordsStartingWith(text, letter)` that returns the list of words starting with `letter` in the string `text`. The result should not be case sensitive, e.g. 'about' should be returned if the letter 'a' or 'A' is given as parameter.

For example using the variable `sampleText` we should obtain:

```
>>> getWordsStartingWith(sampleText, 'a')
['As', 'a', 'about', 'adding', 'a', 'ago', 'a',
'around', 'Amsterdam', 'a', 'and', 'an', 'about',
'a', 'ABC', 'appeal', 'as', 'a', 'a', 'a']
>>> getWordsStartingWith(sampleText, 'z')
[]
```

Hint: You may want to use your solution from question 1.

Question 3: *from a resit paper.*

As you can see in question 2, there are many repetitions of the word 'a' in the list. Improve your solution so no repetition of the same word occurs in the list.

```
>>> getWordsStartingWith(sampleText, 'a')
['As', 'a', 'about', 'adding', 'ago', 'around',
'Amsterdam', 'and', 'an', 'ABC', 'appeal', 'as']
>>>
```

Question 4: *difficult.*

Rather than just removing duplicate words like in question 3, perhaps it would be more interesting to keep track of the number of occurrences of each word. For example in `sampleText`, 'a' occurs 9 times, while 'as' occurs twice (one 'As' and one 'as) and 'python' three times.

Implement `printWordsFrequency(text)` which returns the result of our computation, e.g. for each word in `text`, provide its number of occurrences in `text`. What kind of data structure would you use to represent the result of the computation? Compare your solution with someone else, are they similar? If not, which one is best, and why?