Python for data science SP1 2023/2024 Assignment 2

There are two problems in this assignment.
To submit:

- A Jupyter notebook (.ipynb) with clearly marked solutions for: Problem 1 and Problem 2. The notebook must show examples that you used to run the codes, and the outputs that you get after running the codes.
- An html file (a result of "Save and Export Notebook As" html of your Jupyter notebook) after you run all the codes or a link to your project if you are using Colab or Deepnote.

Do not forget to 1) add your name and your group partner's name at the very top of your notebook and 2) add comments where suitable.

Note: If you add any details or make any ssumptions, describe these in your submission!

Problem 1

Looking for a word in a dictionary is very easy since words are sorted alphabetically. But, what if we are looking for all the words for which the second letter is an *a,* it is impossible unless we sort all the words by their second letter. In this case, we would need a different dictionary for each letter position.
The objective of this exercise is to design and implement a kind of dictionary that will allow you to find all the words with a given letter in a given position.

Tasks:

1. Write the simplest solution that allow you to find all the words with a given letter in a given position.
2. Write a more efficient solution by using a different data structure. Explain and motivate your choice.
3. Compare the execution time between these two solutions.

Write your solutions as functions, where suitable.
Examples are shown below. For this, let's assume that:

- **data_structure** is the data structure you will use for this problem.
- **words** represent a sample of words from a dictionary. You can also use the file *words.txt*.

words = ['class', 'case', 'course', 'dictionary', 'java', 'list', 'program', 'python', 'tuple', 'word']

```
w = words_letter_position(data_structure, 'a', 1)
print("results = ", w)
output:
results = ['case', 'java']
```

```
w = words_letter_position(data_structure, 't', 3)
print("results = ", w)
output:
results = ['dictionary', 'list']
```

Problem 2

An anagram is a word formed by rearranging the letters of a different word by using all the original letters exactly once. For example, the word **teacher** can be rearranged into **cheater** or the word **rebuild** can be rearranged into **builder**.

Tasks:

1. Write a program that reads a word list from the file *words.txt* and as an output it prints to a destination file (the file name is given as an input):
   - all the lists of words that are anagrams ordered by the length of lists.
   - all the lists of words that are anagrams and contains n letters (n is given as an input).

   Note that all anagrams should also be in the *words.txt* file.

Here is an example. For 6 letters, we will have:

...
(2, ['append', 'napped'])
...
(12, ['enters', 'ernest', 'estren', 'nester', 'renest', 'rentes', 'resent', 'sterne', 'streen', 'tenser', 'ternes', 'treens'])
...

2. Which data structure did you use and why?