File Analysis

Starting from the provided `file_analysis.py` template, write a program that processes the two provided text files (“xian_1.txt” and “xian_2.txt”) and produces the following output files.

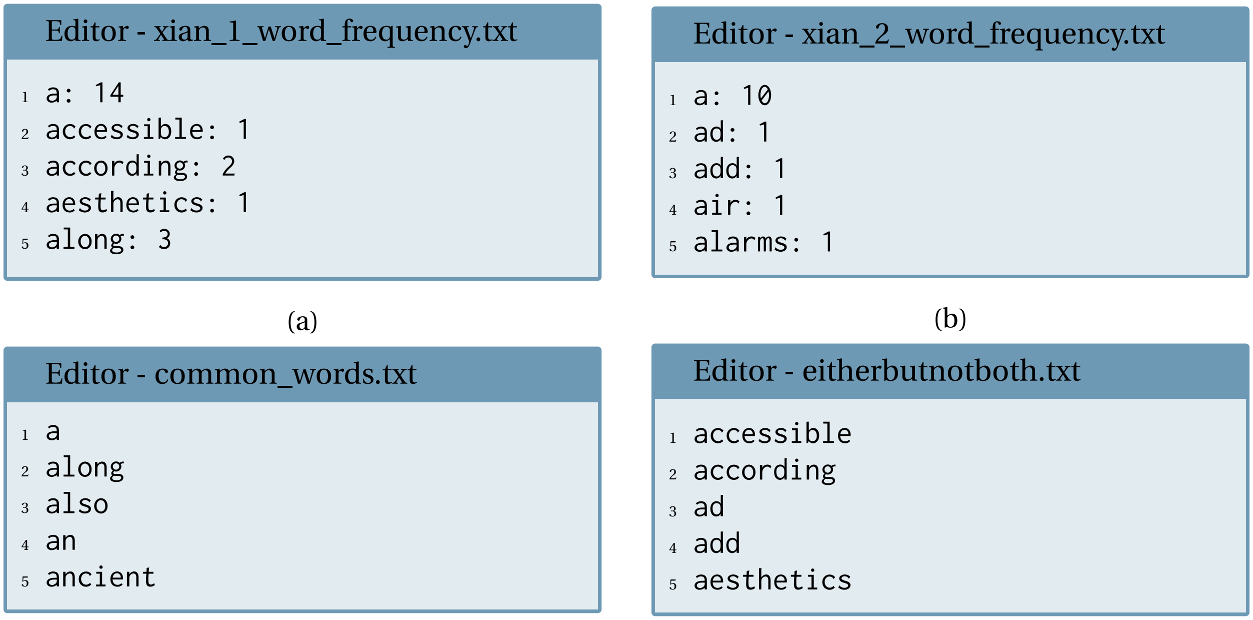
1. `xian_1_word_frequency.txt` This file should contain all of the unique words in the file `xian_1.txt` printed one per line in alphabetical order. Each word should be followed by a colon, a space, and the number of times that word appears in `xian_1.txt`.

2. `xian_2_word_frequency.txt` This file should contain all of the unique words in the file `xian_2.txt` printed one per line in alphabetical order. Each word should be followed by a colon, a space, and the number of times that word appears in `xian_2.txt`.

3. `common_words.txt` This output file should contain all of the words that appear in both `xian_1.txt` and `xian_2.txt`, printed one per line in alphabetical order.

4. `eitherbutnotboth.txt` This file should contain all of the words that appear in either `xian_1.txt` or `xian_2.txt` but no words that appear in both. The words should be printed one per line in alphabetical order.

For comparison purposes, all the words should be converted to lower case and all leading and trailing punctuation should be removed. This first five lines of each of the output files is shown in Figure 1. Save your program as `file_analysis.py` and submit it along with a screenshot showing the first few lines of **all 4** of the output files.



Editor - xian_1_word_frequency.txt

```
a: 14
accessible: 1
according: 2
aesthetics: 1
along: 3
```

(a)

Editor - xian_2_word_frequency.txt

```
a: 10
ad: 1
add: 1
air: 1
alarms: 1
```

(b)

Editor - common_words.txt

```
a
along
also
an
ancient
```

(c)

Editor - eitherbutnotboth.txt

```
accessible
according
ad
add
aesthetics
```

(d)

Figure 1: The first five lines of output files (a) `xian_1_word_frequency.txt`, (b) `xian_2_word_frequency.txt`, (c) `common_words.txt`, and (d) `eitherbutnotboth.txt`.

Hints:

- The `string` module includes a `punctuation` string which contains all of the ASCII punctuation characters.
- It might be helpful to write a function that accepts a filename as an argument and returns a list of all the words in that file. Then have another function that accepts a list of words as an arguement and returns a dictionary with the unique words in the list as keys and the number of times each word appears in the file as values.
- You can use the `sorted()` function to get a sorted list of the keys and values in a dictionary.
- You can use the `set()` function to create a set from the keys in a dictionary.