Sorted: CS Coding Challenge

We ask you to undertake a small programming exercise that consists of

- Part 1: write a program to solve a problem
- Part 2: write a (simple) test suite for your problem
- Part 3: write an analysis of other solutions to the problem

If you use code that is not your own, e.g. a library routine, please indicate its source. The exercise should take about 4 hours, some people might be able to complete the exercise in much less time.

Part 1: write a program to solve a problem

Please write a program that takes a list of strings containing integers and words and returns a sorted version of the list. The goal is to sort this list in such a way that all words are in alphabetical order and all integers are in numerical order. Furthermore, if the nth element in the list is an integer it must **remain** an integer, and if it is a word it must **remain** a word.

In addition, the strings and integers may contain characters that are ascii symbols that neither belong to letter set nor digit set (i.e. "#", "%", ";", etc). You are required to remove them during the process so that the output will contain only letters or digits. For example, if a string is "sym*bo+l", the output should be "symbol". If an integer is "12%3", the output should be "123". You don't have to worry about strings or integers that contain only non-letter-non-digit characters, like "^!?", "&", etc.

For example: `20 cat bi?rd 12 do@g` >> should read >> `12 bird cat 20 dog`

Input and Output

The input for your code will be a file that includes a single, possibly empty, line containing a space-separated list of strings to be sorted. Words will not contain spaces, will contain upper-case, lower-case letters a-z and maybe non-letter-non-digit characters. Integers will be in the range -999999 to 999999, and might also contain non-letter-non-digit characters.

The output must be printed into a file named "result.txt". The content of the file is the list of strings, sorted per the requirements above. Strings must be separated by a single space, with no leading space at the beginning of the line or trailing space at the end of the line.

The program should take the input file name as the first argument, output file as the second argument:

```
root:#  ./listSorting.py <path-to-input-file>/list.txt <path-to-output-file>/result.txt
```

The quality of your code is important; it’s good to learn best practices now! Write a program that is coded in a professional manner and add comments where necessary and helpful!

Part 2: Test Suite

Write a (simple) test suite for your program (and while it’s optional, please do consider completing this section). Tests help you break down your program into its basic functional units that each serve one and one function only. In other words, tests help you develop good programming practice. Also, tests help you discover and handle edge cases. Test driven development is becoming increasingly relevant to data science roles -- it supports good practice and helps you become a better coder. If you’re not familiar with tests, have a look here and here.

Part 3: Analysis Document

In addition to the program and the test suite, write a short analysis of the algorithm you have chosen and other possible algorithms to solve the problem (like, a couple paragraphs is what we’re looking for). Look at the expected running time of the different algorithms as a function of the number of words and the number of duplicate words.