Anagram Assignment

## The Task

Write a program that takes as argument the path to a file containing one word per line, groups the words that are anagrams to each other, and writes to the standard output each of these groups.

The groups should be separated by newlines and the words inside each group by commas.

## The Data

You can make the following assumptions about the data in the files::

* The words in the input file are ordered by size
* The files may not fit into memory all at once (but all the words of the same size would)
* The words are not necessarily actual English words, for example, “abc” and “cba” are both considered words for the sake of this exercise.

The files provided in the `Data` folder are just sample input data to help you reason about the problem. Production files will be much bigger.

If you make other assumptions, make sure you write them down in a readme in your submission

### How much time do I have?

Candidates usually complete the test in 2 to 3 hours, but take as much time as you need. The time taken to complete the test is not taken into consideration during the review.

It's better to spend some more time on the test and submit a complete one, rather than rush it and submit a half-baked one (aka: we can't read your mind yet :)).

### What language should I use?

You choose the language. Our suggestion is to use the language you feel strongest at.

## Our expectations

We expect **Production-Ready** **code**: code that's not only accomplishing the task, but it’s resilient, performing and maintainable by anybody in the team. You should also take into account the assumptions given above.

Details are important, and you should treat your submission as if it were a pull request ready to go live and serve millions of users.

### What should a good submission include?

1. Full suite of automated tests covering the edge cases
2. Exceptions handling
3. Separation of concerns
4. The resources (CPU, memory, disk...) are used efficiently
5. README will describe your solution including:
   1. chosen language,
   2. how we can run your code
   3. Big O analysis
   4. reasons behind data structures chosen
   5. what you would do given more time
6. The code should be easy to read, with well named variables/functions/classes. We like Clean Code principles :)

### **NOTES:**

* We expect your code to compute the anagrams **without the help of any library**
* You are allowed to use any library for any other aspect functional to the task (e.g.: handling the CLI, testing, I/O)
* If CLI is not available in your development stack (e.g.: mobile) you can rely on automated tests and/or UI to feed your program with the input data and verify the output.
* The order of the groups in the output does not matter

Example:

command\_to\_run\_your\_program Data/example1.txt

Output:

abc,bac,cba

unf,fun

hello

## Submission Instructions:

Please wrap your submission into a single zipped file and send it to [codechallenge@superawesome.com](mailto:codechallenge@superawesome.com), as well as ext-joe.rinaldi@superawesome.com.

Any questions - just let us know.

We can’t wait to see what you can do with this!

SA recruitment team