# Anagram Finder Programming Task

## REQUIREMENTS

An anagram is a word, phrase, or name formed by rearranging the letters of another, such as *cinema*, formed from *iceman*.

For this exercise, you’ll write two versions of a program in the programming language(s) you wish that reads a dictionary file and allows a user to find anagrams.

For your first version, focus on getting the job done quick and dirty, but still technically correct (i.e. it doesn’t have to be pretty, but it needs to work). For this version, you may assume that you’ll only need to run this program as a one time thing.

For the second version of your program, aesthetics and efficiency are both important. You may refactor your first version first or start fresh, but assume your program is going to be run to find many anagrams over and over, possibly part of some larger server infrastructure. For this version, we want to see best practices and the program needs to be efficient as possible.

In each case, the program should accept a single parameter which will be the name of the dictionary file. The dictionary file consists of a list of words. Each word is on separate line.

The program should provide a command line prompt where a user can input a word of their choice. On hitting enter the program should find all anagrams, if any exist, of the word and print them out on the next line as a comma separated list. If no anagrams are found it should print out “No anagrams found for <word>”.

Include in the output timings for loading the dictionary and timings for each anagram set request.

The program should continue to prompt for anagrams until “exit” is typed at the prompt.

You may use any classes that are part of the language core library but no external third party libraries.

Consider the performance of your application if it is run against a dictionary with over 100,000 words including longer words than that supplied in the sample dictionary.

## DELIVERABLES

Once you have completed the first version of your program, ensure that it compiles then create a copy or snapshot of the source code in a .zip or .tar archive, including any build instructions as necessary.

DO NOT SUBMIT IT YET.

Complete your second version, ensure that it compiles then create a separate copy or snapshot of the source code in a .zip or .tar archive.

Finally, submit both projects at the same time via the submission link provided with the assignment email. Also provide roughly how long the task took you.

## EXAMPLE OUTPUT

MacBookPro: ~$ java AnagramFinder dictionary.txt

Welcome to the Anagram Finder

-----------------------------

Dictionary loaded in 250 ms

AnagramFinder>stop  
4 Anagrams found for stop in 12ms  
pots,opts,stop,tops

AnagramFinder>accept  
No anagrams found for accept in 9ms

AnagramFinder>exit  
MacBookPro: ~$