Back-End Coding Exercise — 2–3 hours

The exercise is to write a command line driven text search engine in the language you feel most comfortable with.

Examples of the usage would be:

> **python simple_search.py** *<pathToDirectoryContainingTextFiles>*
>
> or
>
> **java** -jar *<jarName> <mainClassFile> <pathToDirectoryContainingTextFiles>*

This should read all the text files in the given directory, building an *in-memory* representation of the files and their contents, and then give a command prompt at which interactive searches can be performed.

An example session might look like:

```
$ java -jar SimpleSearch.jar Searcher /foo/bar
14 files read in directory /foo/bar
search> to be or not to be
file1.txt:100%
file2.txt:90%
search>
search> cats
no matches found
search> :quit
$
```

The search should take the words given on the prompt and return a list of the top 10 (maximum) matching filenames in rank order, giving the rank score against each match.

Note: We would like to see a working console application, please focus your attention on the search algorithm and the basic console functionality (you can assume that the input strings are sane).

Ranking

- The rank score must be 100% if a file contains all the words
- It must be 0% if it contains none of the words
- It should be between 0 and 100 if it contains only some of the words, but the exact ranking formula is up to you to choose and implement

Expectations (please read carefully)

- **The program must run and return results on search queries**
- **The code must include unit tests and be well structured to support future maintenance and evolutions.** Try to use the best practices that you have learned from your past experiences
- You should be spending 2-3 hours on the task, but approaching 'production like' quality is more important than implementation speed and feature completeness
- External libraries are forbidden, other than test/mocking frameworks
- A build procedure whenever applicable (compiled language)

Things to consider in your implementation

- Ranking score design — **start with something basic**, then iterate as time allows
- The in memory representation — searches should be relatively efficient
- What constitutes a word?
- What constitutes two words being equal (and matching)?
- You will be discussing your code in the next stage of the interview process with our engineers, so keep in mind how your code can be iterated on, how it can be maintained and what features could be added later

Deliverables

- Implementation, test code and compiled program for the above
- A README file containing instructions so that we know how to build and run your code

An example starting point is on the next page

Example starting point in Java. This may not be as testable as you would want.

```
import java.io.File;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        if (args.length == 0) {
            throw new IllegalArgumentException("No directory given to index.");
        }

        final File indexableDirectory = new File(args[0]);
        //TODO: Index all files in indexableDirectory
        try (Scanner keyboard = new Scanner(System.in)) {
              while (true) {
                  System.out.print("search> ");
                  final String line = keyboard.nextLine();
                  // TODO: Search indexed files for words in line
              }
        }
    }
}
```