**Software engineering home assignment**

The purpose of this task is demonstrating software engineering & coding skills.

We know this task might be time consuming and we respect your time, but try to submit a complete task (even if it takes few days more than expected).

If you have any question don’t hesitate to ask.

Your program should accept four parameters, either in a configuration file or as command line arguments:

1. The location of one input file
2. The location of second output file
3. The URLs for 2 HTTP endpoints

The input file can be of any length, but its lines are guaranteed to be reasonably short (Let's say up to 128 characters), and separated by a newline character. For example:

Raising

Skinny

Elephants

Is

Utterly

Boring

For each line in the input file, you are required to send its content to both HTTP endpoints. Each endpoint will respond with its own string, which is also guaranteed to be reasonably short (Let's say up to 128 characters). If the strings match, you need to write a line with the string "true" to the output file. If they don't, write "false". **The lines in the output file must match the lines in the input file!** So, if the 5th line in the input file got the same response from both endpoints, then the 5th line in the output file must read "true".

While this is easy enough to do in a blocking loop, that would be very slow. Therefore, you are requested to parallelize your HTTP requests in a nonblocking fashion, such that many (Let's say, up to 128 per endpoint) requests can be made of each endpoint at the same time. Nonblocking means that making 128 parallel requests should not require 128 threads.

You are free to use Scala and/or Java to write this program. You are free to use any library available on a major public repository (such as Maven Central). You are free (and required) to use a build tool of your choice, such as SBT, Gradle, or Maven.

Your solution should be delivered to us in the form of a zip or tar archive of the following:

1. Scala and/or Java source files
2. Any additional resources necessary to run (e.g. configuration files)
3. A build script
4. A readme file containing the command we should use to build and run your solution

**Important**:

To help you get started, we provide you with a simple HTTP server application for testing purposes. You are not required to use it, but it will help you test yourself:

1. The application requires Java 8 or above to run
2. You can run it with the following command: **java -jar -Dseed=XXXX -Dport=YYYY iguazio-assignment-http-server.jar**
3. make sure to launch 2 endpoint with different seed (and port of course)
4. When you launch the server, it will listen to: **http://localhost:YYYY** (YYYY according to the port you pass in the command)
5. In order to check that the server is up you can run: **curl --data "hello"** [**http://localhost:YYYY**](http://localhost:YYYY)(YYYY according to the port you pass in the command)

**GoodLuck!**

**Iguazio BD team**

Appendix A: Example input/output

Input file

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

Raising

Skinny

Elephants

Is

Utterly

Boring

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

Endpoint 1 responses

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

1

2

3

4

5

6

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

Endpoint 2 responses

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

2

2

2

2

2

2

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

Output file

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

false

true

false

false

false

false