

# Big Data Algorithms

## Lab-0

Due: 24.10.2021

1. (1pts) **Scala *Hello, World!***

Visit the Scala web page and follow the instructions to install Scala 3. Create the simple application printing the *Hello, world!* text into the console using:

- [compiled Scala \(basic tutorial\)](#),
- [Scala REPL interpreter \(REPL details\)](#),
- Worksheets (worksheet details; Hint: for Visual Studio Code and Metal - extensions will not start if directory is not detected as proper project, i.e. contain the *build.sbt* file which can be empty).

Scala has a documentation - read basic chapter about variables, types, conditionals etc. You will need basics to work through next exercises.

2. (1pts) **Word cloud - book**

- (a) Find some interesting book in simple *.txt* format (example books source).
- (b) Load the book into Scala and divide it into words (remove all punctuation).
- (c) Remove all stopwords (stopwords list).
- (d) Create the collection of pairs in a form of *(word, count)*, which for each *word* will keep the *count* of times this word appeared in the list of words.
- (e) Display some of the most frequently appearing words and create word cloud from them (word cloud generator).

3. (2pts) **Word cloud generator**

This task will extend the book-based words analyzer from previous exercise for more complex data handling - you task will be to wrap it into class which will store all loaded pairs of *(word, count)* and provide the following interface to extend and analyze the collected data:

- (a) Text input - class should be able to receive pure String and perform operations from previous exercise (i.e. division into words, stopwords removal) and finally add the results to currently held context.
- (b) File input - same as above, but for file to be first loaded as a String.
- (c) Word cloud output to console - print selected number of most frequent words.
- (d) Word cloud output to file - write selected number of most frequent words into *.csv* file for loading it to online word cloud generator (with weights).

Finally, prepare the user-friendly console interface (looped, so the user can use it to feed the generator a lot of data).

4. (1pts\*) **Spark word cloud generator**

Bonus task, convert the previously created generator to use Apache Spark framework.