**TestDevLab C++ coding skills test**

**Extended text file processing**

Create a command line application that analyzes text from a given file (passed as a command line parameter ***--in <filename>***) and outputs statistics of the text file.

The application should be able to provide the following statistics:

* Word occurrence: a list of words and the count of the word’s occurrences in the text file, sorted by the most frequent words in descending order (if the frequency of two or more words is equal, sorting should be done in alphabetical order);
* Paragraph count: the number of paragraphs in the text file;
* Sentence count: the number of sentences in the text file;
* Punctuation mark occurrence: a list of used punctuation marks (periods, commas, question marks, ellipses etc.) and their occurrence, sorted in descending order;

The mode of operation for the application should be configurable, that is, the end user should be able to choose in what configuration does the program operate by passing a

***--mode <mode>*** parameter from the command line. The following modes should be defined:

* ***all*** (default mode): output all statistics (word occurrence, paragraph count, sentence count, punctuation mark occurrence);
* ***words***: output only word occurrence statistics;
* ***paragraphs***: output only paragraph count;
* ***sentences***: output only sentence count statistics;
* ***punctuations***: output only punctuation mark occurrence statistics.

Optionally, the application should also provide results output to a text file (with the command line option ***--out <filename>****)*.

Some additional notes on the solution:

* Unit tests must be added that test the correctness of the solution using any of the popular C++ unit testing frameworks (Catch, Boost.Test, Google Test). Provide comments on the testing strategy and the tests that were introduced;
* Usage of the CMake build system generator for creating executable targets and linking external libraries is recommended (but not required);
* Add comments explaining the methods and solutions used in the application.

Example input file:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo sapien, euismod a quam pulvinar, tristique ultrices metus.  
  
Cras vitae ipsum nec turpis sagittis lobortis. Quisque urna ligula, ullamcorper ut mollis quis, convallis sit amet orci.

Example output:

$ ./text-file-processor --in lorem\_ipsum.txt --mode all  
Word count statistics:  
amet -> 2  
ipsum -> 2  
sit -> 2  
a -> 1  
adipiscing -> 1  
consectetur -> 1  
convallis -> 1  
cras -> 1  
dolor -> 1  
donec -> 1  
elit -> 1  
euismod -> 1  
justo -> 1  
ligula -> 1  
lobortis -> 1  
lorem -> 1  
metus -> 1  
mollis -> 1  
nec -> 1  
orci -> 1  
pulvinar -> 1  
quam -> 1  
quis -> 1  
quisque -> 1  
sagittis -> 1  
sapien -> 1  
tristique -> 1  
turpis -> 1  
ullamcorper -> 1  
ultrices -> 1  
urna -> 1  
ut -> 1  
vitae -> 1  
  
Paragraph count: 2  
  
Sentence count: 4  
  
Punctuation mark statistics:  
, -> 5  
. -> 4