10th Practical Class – String Algorithms

Instructions

- Download the zipped file **TP10_unsolved.zip** from the course's Moodle area and unzip it. It contains a .cpp file where you should implement the methods specified below, as well as other .cpp file with Unit tests.
- In the CLion IDE, open the project used in the previous lessons and add the folder TP10, selecting the folder that contains the files mentioned in the previous bullet point.
- Update the *CMakeLists.txt* file by copying, pasting, and adapting the three lines of code of TP10: file, add_executable and target_link_libraries.
- Do "*Load CMake Project*" over the file *CMakeLists.txt*
- Run the project (**Run**)
- You should implement the exercises following the order suggested.
- Implement your solutions in the matcher.cpp file.
- Important note: in case you need to read text files in I/O mode, you should tell CLion where such files are, by redefining the IDE environment variable "Working Directory", through menu Run > Edit Configurations… > Working Directory.

Exercises

1. After searching for documents which include a certain expression (`toSearch`), there is the need to sort them by relevance (number of times the expression occurs). Implement the function:

```
int numStringMatching(string filename, string toSearch)
```

This function returns the number of occurrences of the `toSearch` string in the file named `filename`.

You should read the file line by line, and for each line count the number of times the expression occurs (`kmpMatcher`). The function should return the sum of the values for all lines.

2. After searching for documents which include words similar to a given one (`toSearch`), there is the need to sort them by relevance (average distance). Implement the function:

```
float numApproximateStringMatching(string filename, string toSearch)
```

This function returns the average distance of words in the file named `filename` to the searched for expression (`toSearch`).

For each word in the file, the function should calculate the distance to the searched for word (`editDistance`). It should then return the average distance (sum of distances / number of words).