Word comparison

Submission deadline:	**2021-12-05 23:59:59**
Late submission with malus:	**2022-01-02 23:59:59** (Late submission malus: 100.0000 %)
Evaluation:	**3.0000**
Max. assessment:	**3.0000** (Without bonus points)
Submissions:	1 / 20 Free retries + 10 Penalized retries (-10 % penalty each retry)
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)

The task is to develop function (just a function, not a whole program), which compares two strings.

The required function has the following interface:

```
int sameWords ( const char * a, const char * b );
```

The parameters are two strings to compare. The parameters are read-only (note the `const` qualifier).

A program that passes all mandatory and optional tests with 100 % result may be submitted to code review (i.e., you do not need to pass bonus tests for code review).

The return value is either 1 (match) or 0 (no match). The strings match if they are composed of the same words. That is, each word in the first string is at least once present in the second string and vice versa. Case insensitive comparison is used when comparing the words.

Words in this assignment are composed of characters where function `isalpha` returns nonzero. All other characters are considered word separators.

Submit a source file with the implementation of the required function `sameWords`. Further, the source file must include your auxiliary functions which are called from `sameWords`. The function will be called from the testing environment, thus, it is important to adhere to the required interface. Use the sample code below as a basis for your development, complete `sameWords` and add your required auxiliary functions. There is an example `main` with some test in the sample below. These values will be used in the basic test. Please note the header files as well as `main` is nested in a conditional compile block (`#ifdef`/`#endif`). Please keep these conditional compile block in place. They are present to simplify the development. When compiling on your computer, the headers and `main` will be present as usual. On the other hand, the header and `main` will "disappear" when compiled by Progtest. Thus, your testing `main` will not interfere with the testing environment's `main`.

Your function will be executed in a limited environment. There are limits on both time and memory. The exact limits are shown in the test log of the reference. The time limits are set such that a correct implementation of the naive solution passes all mandatory tests. Thus, the solution may be awarded nominal 100% percent. The algorithm must be improved to pass the bonus test. There are long strings with many words being tested in the bonus test.

The implementation must use C strings. C++ strings (`std::string`) and STL is forbidden. If used, the program will not compile.

The attached archive contains an outline of the implementation and a few test cases.

Sample data: **Download**

✓ Reference

- **Evaluator: computer**
  - Program compiled
  - Test 'Zakladni test podle ukazky': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test meznich hodnot': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.000 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi daty': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.245 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi daty + test prace s pameti': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.008 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi daty (dlouhe retezce, bonusovy)': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.216 s (limit: 1.000 s)
    - Bonus test - success, evaluation: 120.00 %
  - Overall ratio: 120.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.20)

- Total percent: 120.00 %
- Early submission bonus: 0.30
- Total points: 1.20 * (3.00 + 0.30) = 3.96

SW metrics:		Total	Average	Maximum	Function name
	Functions:	**3**	--	--	--
	Lines of code:	**70**	**23.33 ± 14.70**	**41**	`parseWords`
	Cyclomatic complexity:	**17**	**5.67 ± 4.64**	**12**	`parseWords`

1	**2021-12-03 22:57:01**	**Download**
Submission status:	Evaluated	
Evaluation:	3.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample input data': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Borderline test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.000 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.727 s (limit: 2.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test + mem usage test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.013 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test (long strings, bonus)': Abnormal program termination (Time limit exceeded)
    - Cumulative test time exceeded, killed after:: 1.005 s (limit: 1.000 s)
    - Bonus test - failed, evaluation: No bonus awarded
  - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00)
- Total percent: 100.00 %
- Total points: 1.00 * 3.00 = 3.00

SW metrics:		Total	Average	Maximum	Function name
	Functions:	**5**	--	--	--
	Lines of code:	**54**	**10.80 ± 3.71**	**16**	`AllWordSearching`
	Cyclomatic complexity:	**14**	**2.80 ± 0.98**	**4**	`AllWordSearching`