CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

Lab 3B - Strings Test Suite

Home | HW Sets | Schedule | Files | Help | Links

This lab is due by Tuesday, October 10, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

Concepts

This assignment introduces you to a simple form of "unit testing" as a mechanism for exploring the string API. Focus on learning what unit tests are, what an "API" is, and what you can do with string objects.

APIs (Application Programming Interface)

As you can see above, the acronym API stands for Application Programming Interface. But what does *that* mean? In a nutshell, an API describes what you can do with a particular library or object that you are provided (or that you create). It describes how your code can "interface with" or "use" a particular library or object.

For example, the string API consists of member functions that tell you how long the string is, allow you to capitalize the string, tell you whether it contains a specific character, or allow you to extract a part of the string.

RMP-H

While the API really is the programmatic components that you can actually use, we often rely on *API documentation* to discover what you can do with a particular library or object. For example, in this lab you will *use* the string API, and you will need to look up some API documentation about how you can use string objects.

Unit Testing

As entire books have been written on unit testing, we will merely introduce the topic here. A "unit test" is a small piece of code designed to test a specific part of a program's functionality. In other words, they are bits of code that test the functionality of other code!

For this lab, that's all the preliminary information you really need to know about unit testing. You will actually write the unit tests, eventually getting the entire test suite to pass (at which point you should go outside and run a victory lap).

Instructions

A skeletal test suite (a collection of functions) has been provided for you in **string_tests.zip**. Notice that the functions in this code are defined across multiple files. The functions are grouped into files based on their application. In this case, each string function is performing a single test on a string. The function will need to return the result of the test.

When you first run the program, you will see the following output:

```
Testing your functions...

Test #  1                    Testing string_length():     PASSED
Test #  2                    Testing string_length():     PASSED
Test #  3                    Testing string_length():     PASSED
Test #  4                    Testing string_length():     PASSED

Test #  5                   Testing string_char_at():  !!>FAILED<!! Returned: "" != Expe
Test #  6                   Testing string_char_at():  !!>FAILED<!! Returned: "" != Expe
Test #  7                   Testing string_char_at():  !!>FAILED<!! Returned: "" != Expe

Test #  8                    Testing string_append():  !!>FAILED<!! Returned: "There's a
Test #  9                    Testing string_append():  !!>FAILED<!! Returned: "It's the 
Test # 10                    Testing string_append():  !!>FAILED<!! Returned: "" != Expe
Test # 11                    Testing string_append():     PASSED

Test # 12                    Testing string_insert():  !!>FAILED<!! Returned: "If you ca
Test # 13                    Testing string_insert():  !!>FAILED<!! Returned: "carefully
Test # 14                    Testing string_insert():  !!>FAILED<!! Returned: "" != Expe
Test # 15                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp
Test # 16                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp
Test # 17                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp
Test # 18                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp
Test # 19                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp
Test # 20                       Testing string_find():  !!>FAILED<!! Returned: "0" != Exp

Test # 21                  Testing string_substring():  !!>FAILED<!! Returned: "Such a na
Test # 22                  Testing string_substring():  !!>FAILED<!! Returned: "Such a na
Test # 23                  Testing string_substring():  !!>FAILED<!! Returned: "Such a na
Test # 24                  Testing string_substring():  !!>FAILED<!! Returned: "Something
Test # 25                  Testing string_substring():  !!>FAILED<!! Returned: "Something
Test # 26                  Testing string_substring():     PASSED

Test # 27                    Testing string_replace():  !!>FAILED<!! Returned: "Strings a
Test # 28                    Testing string_replace():  !!>FAILED<!! Returned: "Show me t
Test # 29                    Testing string_replace():     PASSED
Test # 30                    Testing string_replace():     PASSED

Test # 31                 Testing string_first_word():  !!>FAILED<!! Returned: "The quick
Test # 32                 Testing string_first_word():  !!>FAILED<!! Returned: "A man a p
Test # 33                 Testing string_first_word():  !!>FAILED<!! Returned: "I have th
Test # 34                 Testing string_first_word():  !!>FAILED<!! Returned: "Testing s
Test # 35                 Testing string_first_word():     PASSED
Test # 36                 Testing string_first_word():  !!>FAILED<!! Returned: "Uh oh" !=
Test # 37                 Testing string_first_word():     PASSED

Test # 38          Testing string_remove_first_word():  !!>FAILED<!! Returned: "The quick
Test # 39          Testing string_remove_first_word():  !!>FAILED<!! Returned: "Testing s
Test # 40          Testing string_remove_first_word():  !!>FAILED<!! Returned: "Goodbye"
Test # 41          Testing string_remove_first_word():     PASSED
Test # 42    Testing string_remove_first_word() twice:  !!>FAILED<!! Returned: "The quick
Test # 43    Testing string_remove_first_word() twice:  !!>FAILED<!! Returned: "Goodbye"

Test # 44                Testing string_second_word():  !!>FAILED<!! Returned: "The quick
Test # 45                Testing string_second_word():  !!>FAILED<!! Returned: "A man a p
Test # 46                Testing string_second_word():  !!>FAILED<!! Returned: "Testing s
Test # 47                Testing string_second_word():  !!>FAILED<!! Returned: "I have th
Test # 48                Testing string_second_word():  !!>FAILED<!! Returned: "Single" !
Test # 49                Testing string_second_word():  !!>FAILED<!! Returned: "Uh oh" !=
Test # 50                Testing string_second_word():     PASSED

Test # 51                 Testing string_third_word():  !!>FAILED<!! Returned: "The quick
Test # 52                 Testing string_third_word():  !!>FAILED<!! Returned: "A man a p
Test # 53                 Testing string_third_word():  !!>FAILED<!! Returned: "I have th
Test # 54                 Testing string_third_word():  !!>FAILED<!! Returned: "Single" !
Test # 55                 Testing string_third_word():  !!>FAILED<!! Returned: "Uh oh" !=
Test # 56                 Testing string_third_word():     PASSED

Test # 57                  Testing string_nth_word(1):  !!>FAILED<!! Returned: "The quick
Test # 58                  Testing string_nth_word(2):  !!>FAILED<!! Returned: "The quick
Test # 59                  Testing string_nth_word(3):  !!>FAILED<!! Returned: "The quick
Test # 60                  Testing string_nth_word(4):  !!>FAILED<!! Returned: "The quick
Test # 61                 Testing string_nth_word(5):  !!>FAILED<!! Returned: "The quick
Test # 62                 Testing string_nth_word(6):  !!>FAILED<!! Returned: "The quick
Test # 63                 Testing string_nth_word(7):  !!>FAILED<!! Returned: "The quick
Test # 64                 Testing string_nth_word(8):  !!>FAILED<!! Returned: "The quick
Test # 65                 Testing string_nth_word(9):  !!>FAILED<!! Returned: "The quick
Test # 66                Testing string_nth_word(10):  !!>FAILED<!! Returned: "The quick
Test # 67                Testing string_nth_word(11):  !!>FAILED<!! Returned: "The quick
Test # 68                 Testing string_nth_word(1):     PASSED
Test # 69                 Testing string_nth_word(3):     PASSED

Test # 70                  Testing string_tokenize():  !!>FAILED<!! Returned: {} != Expe
Test # 71                  Testing string_tokenize():  !!>FAILED<!! Returned: {} != Expe
Test # 72                  Testing string_tokenize():  !!>FAILED<!! Returned: {} != Expe

Test # 73                  Testing string_tokenize():  !!>FAILED<!! Returned: {} != Expe

Test # 74                Testing string_substitute():  !!>FAILED<!! Returned: "The Gxxgl
Test # 75                Testing string_substitute():  !!>FAILED<!! Returned: "$chool of
Test # 76                Testing string_substitute():     PASSED
Test # 77          Testing string_substitute() twice:  !!>FAILED<!! Returned: "D--" != E

Test # 78                  Testing string_to_lower():  !!>FAILED<!! Returned: "This SHOU
Test # 79                  Testing string_to_lower():  !!>FAILED<!! Returned: "MNASDF874
Test # 80                  Testing string_to_lower():  !!>FAILED<!! Returned: "C++" != E
Test # 81                  Testing string_to_lower():     PASSED
Test # 82                  Testing string_to_lower():     PASSED

Test # 83                  Testing string_to_upper():  !!>FAILED<!! Returned: "This SHOU
Test # 84                  Testing string_to_upper():  !!>FAILED<!! Returned: "mnasdf874
Test # 85                  Testing string_to_upper():  !!>FAILED<!! Returned: "c++" != E
Test # 86                  Testing string_to_upper():     PASSED
Test # 87                  Testing string_to_upper():     PASSED

Test # 88                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 89                   Testing string_compare():     PASSED
Test # 90                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 91                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 92                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 93                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 94                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 95                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 96                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 97                   Testing string_compare():     PASSED
Test # 98                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test # 99                   Testing string_compare():  !!>FAILED<!! Returned: "0" != Exp
Test #100                   Testing string_compare():     PASSED

Tests Passed:  23 / 100 (23%)

Not all tests are passing, errors remain...
```

Your job: complete each TODO using the string API such that all tests pass. You should not modify the contents of `run_all_tests()` in this lab, and instead only insert code at each TODO statement. When you have finished the task, remove the TODO comment.

You should start by reading the body of the function called `run_all_tests()` in order to see what your functions must accomplish. For example, the function `string_length()` must return the length of the string "Now" using the string API. Take a look at the function `string_length()` to see an example of a successful implementation.

When your program prints `PASSED` for a given unit test instead of `FAILED`, then you know that your function implementation for that test is complete.

Hints

- We recommend that you complete the functions in the order in which they are called in `run_all_tests()` .
- If you have trouble getting a function to pass its test, use the debugger with breakpoints to help you troubleshoot what your code is doing.
- Leverage the string API as much as you can. Explore the **string API** to see how the functions work. The documentation may be somewhat confusing; thus, if there is something you don't understand, be sure to ask!

Functional Requirements

- You must not modify the contents of `main()` or `run_all_tests()` .

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Tuesday, October 10, 2023, 11:59 PM
0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `string_functions.cpp` files and name the zip file `L3B.zip`. Upload this zip file to Canvas under L3B.

This lab is due by Tuesday, October 10, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

[Jump to Top] [Site Map]