CMSC 216 Exercise #2 Summer 2022

Text Manipulation **Due: Wednesday, Jun 15, 2020, 11:55 pm**

1 Objectives

To practice strings and pointers.

2 Overview

The first thing you need to do is to copy the directory text_manipulation we have left in the grace cluster under the exercises directory. Remember that you need that folder as it contains the .submit file that allows you to submit.

3 Specifications

For this exercise you will implement two functions that manipulate text. The prototypes for the functions can be found in the text_manipulation.h file.

1. **int remove_spaces(const char *source, char *result, int *num_spaces_removed);**

   This function places a copy of the **source** string in the out parameter **result** where all leading and trailing spaces have been removed. If the out parameter **num_spaces_removed** is different than NULL, the function will set the integer associated with the parameter to the number of spaces removed. The function will return one of two values: FAILURE or SUCCESS (see file *text_manipulation.h*).

   a. FAILURE - if the source string is NULL or its length is 0. In this case the **result** string is not assigned a new value (it keeps its original value).
   b. SUCCESS - if spaces can be removed or no spaces are present.

2. **int center(const char *source, int width, char *result);**

   This function generates a new string into the **result** out parameter where the **source** input string has been centered in a string with length specified by the **width** parameter. You will center the string by adding (to the left and right of the original string) a number of spaces that corresponds to (width - source string length) / 2. Notice that the resulting centered string will have a length that is less than width when (width - source string length) is odd. For example, if we were to center "dogs" in a field with of 7, the resulting string will be " dogs " (1 space to the left, 1 space to the right). The function will return one of two values: SUCCESS or FAILURE (see file *text_manipulation.h*).

   a. FAILURE - if source is NULL, if source length is 0, or if the width is less than the source length.
   b. SUCCESS - if item is centered.

You should look at the public tests in order to understand the functionality associated with the functions you must implement. For this exercise you can assume the user will not provide a string containing only blank characters. In addition if a string has multiple words, the spaces between those words must be preserved.

4 Requirements

1. Unlike other assignments for this course, you can work together with other classmates, but you may NOT exchange any code.

2. If you are having problems with your code, use the class debugging guide available at:
   http://www.cs.umd.edu/~nelson/classes/resources/cdebugging/
3. Your grade will be based on the results obtained from the submit server. It is your responsibility to verify that your program generates the expected results on the submit server.
4. Your code must be written in the file text_manipulation.c.
5. Do not add a main function to the text_manipulation.c file.
6. To compile a public test, compile your text_manipulation.c file along with a public test file. For example:
   gcc public01.c text_manipulation.c
7. One source of bugs is forgetting to add a null character to a string.
8. All your C programs in this course should be written using the compiler `gcc` with the options defined at
   http://www.cs.umd.edu/~nelson/classes/resources/setting_gcc_alias/
9. Your program should be written using good programming style as defined at
   http://www.cs.umd.edu/~nelson/classes/resources/cstyleguide/
10. You just need to implement the two functions described above. You may write additional functions if you want, but define them as static.
11. You may not change the text_manipulation.h file provided.
12. You are encourage to define your own tests (similar to public01.c, public02.c, etc.).
13. You can use the C string library (string.h).
14. You may not use any functions from ctype.h.
15. You can assume the result char array will be long enough to contain the string functions will create.
16. You may not use dynamic memory allocation.
17. The example ptr_add_sub_overview.c reviews concepts associated with this exercise.

5 Submitting your assignment

In the assignment directory (text_manipulation) execute the command **submit**.

6 Grading Criteria

Your assignment grade will be determined with the following weights:

Results of public tests	20%
Results of release tests	80%

Even though we will not look at your code, we expect good style in your code.

7 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. We take academic integrity matters seriously. Please do not post assignment solutions online (e.g., Chegg, github) where others can see your work. Posting code online can lead to an academic case where you will be reported to the Office of Student Conduct.