Lab 7 - C Structs: Pointers and Functions. Strings

General Requirements

Your functions must not be recursive. Repeated actions must be implemented using C's `while`, `for` or `do-while` loop structures.

None of the functions you write should perform console input; for example, contain `scanf` statements. None of your functions should produce console output; for example, contain `printf` statements.

You must format your C code so that it adheres to one of two commonly-used conventions for indenting blocks of code and placing braces (K&R style or BSD/Allman style). Instructions were given in the VS Code installation instructions.

Finish each exercise (i.e., write the function and verify that it passes all its tests) before you move on to the next one.

Getting Started

Step 1: Depending on your operating system, download (**Lab7_win.zip** or **Lab7_mac.zip**) the starting code from Brightspace and extract the zipped file.

Step 2: Double click on Lab7.code-workspace

Step 3: lab6_xxx folder contains the following files:

- struct_string.c contains incomplete definitions of several functions you have to design and code.
- struct_string.h contains the declaration of the `fraction_t` struct, `grade_t` struct and the function prototypes for the functions you will implement. **Do not modify struct_string.h.**
- main.c implement a *test harness* (functions that will test your code, and a `main` function that calls these test functions). **Do not modify `main` or any of the test functions.**

Step 4: Run the project. It should build without any compilation or linking errors.

Step 5: Execute the project. For the functions where the test harness is provided (the functions in main.c) will report several errors as it runs, which is what we'd expect, because you haven't started working on the functions.

Step 6: Open struct_string.c in the editor. Do Exercises 1 through 4. Remember, you must not make any changes to main.c or struct_string.h All the code you write must be in struct_string.c.

Exercise 1

In this exercise, you are going to rewrite the functions that you wrote in Lab 6, so that all your functions use pointers to fraction_t objects, i.e. call-by-reference, rather than passing structures by value. Note that you will **not** update your Lab 6 files. You will write the new functions in Exercise 1 of your Lab 7 solution.

Copy your gcd function from Lab 6 to Lab 7. No changes are needed for this function.

The other five functions in Lab 7 exercise 1, all need to be updated from your Lab 6 versions as per the new signatures provided.

Check that the output for exercise 1 (the fraction functions) is now correct. In addition, double-check that you have followed the instructions above and in the function descriptions.

Exercise 2

In this exercise, we have an array of student data. Each element of the array is a student_t struct. This struct contains an array of grade_t structs. Thus, we have an array of structs, where each struct contains another array of structs.

Your task is to write two functions: update_gpa and calc_gpa. Each function has one input parameter: a pointer to one of these structs.

The first one, update_gpa is very short and basically consists of calling calc_gpa for each student.

The function calc_gpa is a bit more complex. It calculates the GPA for each student. If you are not familiar with how to calculate a GPA, here is the information: https://carleton.ca/npsia/program-hub/how-to-calculate-your-gpa/.

Check that the output for Exercise 2 now includes the correct GPAs. In addition, double-check that you have followed the instructions above and in the function descriptions.

Exercise 3

In this exercise, you are going to write the function count_char that takes two parameters a string and a char. The function counts and returns how many times the char appears in the string.

For example

count_char("Hello world ",'l') should return 3

count_char("Hello world ",'a') should return 0

Your function must use array-indexing notation to iterate over the characters of the string.

Exercise 4

In this exercise, you are going to rewrite the function count_char from exercise 3 to use a walking pointer. The function has the following signature.

```
int count_char_wp(const char *, char);
```

Wrap-up

Submit struct_string.c to Brightspace.

Before submitting your lab work:

- Review *Important Considerations When Submitting Files to Brightspace* (lab3 file). Remember: submit early, submit often.
- Make sure that struct_string.c has been formatted to use K&R style or BSD/Allman style, as explained in *General Requirements*.
- Ensure you are submitting the file that contains your solutions, and not the unmodified file you downloaded from Brightspace!

You are permitted to make changes to your solutions and resubmit the files as many times as you want, up to the deadline. You need to demo your work to TAs during the lab for grades.