CSC 111 - FALL 2023
FUNDAMENTALS OF PROGRAMMING WITH ENGINEERING APPLICATIONS
ASSIGNMENT 3
UNIVERSITY OF VICTORIA

Due: Sunday, Oct. 8th, 2023 at 11:59pm. **Late assignments will not be accepted.**

This assignment will be submitted electronically through BrightSpace (as described in 'Submission Instructions' below). All code submissions must be your own work. Sending or receiving code from other students is plagiarism.

1 Overview

This assignment covers control flow, including if-statements and functions, as well as the use of various arithmetic operators. It also provides an introduction to **error handling** in cases where a program component receives incorrect data, and to **unit testing** to validate the correctness of a set of functions.

You will implement a set of functions which perform various operations on calendar dates. A template file `date_functions.c` has been provided containing several empty functions (along with a `main()` function containing some simple tests). Your task is to implement each function according to the specification given in the comments above the function. You may also add extra functions as needed, but you are not permitted to change the function signature of the existing functions **in any way**. For example, you may not rename the functions, change the types or names of the parameters, add new parameters or modify the return type. Submissions which violate this requirement will receive a mark of zero.

Although each function will be marked separately, you must ensure that the **entire file** compiles successfully (since a syntax error in one function will make it impossible for the teaching team to compile your submission and mark any of your functions).

You are responsible for thoroughly testing each function. Some basic tests have been provided in the `main()` function of the template, but you will need to do more testing to fully validate your implementations. When the code is tested, we will substitute our own `main()` function to test your code. Note that the contents of `main()` in your submission **will not be marked** (although, as mentioned above, you must ensure that the entire file compiles successfully, including whatever code you put in `main()`); all of the marks for this assignment are based on the other functions.

The sections below describe each function in more detail. In the event that this document is not consistent with the specifications written in the posted template code, the template code will be considered correct.

2 Your Task

Download the template `date_functions.c` and implement all of the empty functions according to their specification.

Once complete, your program must compile and run in our virtual lab environment (`https://jhub.csc.uvic.ca`). It must be possible to compile your program using the command

```
gcc -Wall -std=c18 -o date_functions date_functions.c
```

and to run your program (after compiling successfully) using the command

```
./date_functions
```

A full specification (and function signature) for each required function is already present in the starter code posted on BrightSpace. You **must** implement the function according to the provided specification and you are not permitted to modify the function signature in any way. You are permitted to add extra functions as needed.

- `is_leap_year` (1 mark)
- `days_in_year` (1 mark)
- `days_in_month` (3 marks)
- `date_valid` (4 marks)
- `day_index` (3 marks)
- `chronological_order` (3 marks)
- `weekly_reminders` (5 marks)

You may notice that the functionality provided by some functions is helpful for other functions. For example, if you call `is_leap_year` in the `days_in_year` or `days_in_month` function, the implementation of those two functions will be significantly shorter and easier to write. This assignment has been designed to be much easier if you effectively reuse functionality by calling some of the earlier functions inside the later functions.

The sections below contain some clarifications and suggestions for some aspects of the assignment.

2.1 Testing Leap Years

The `is_leap_year` function tests if a particular year is a leap year. You must use the following algorithm (or something which produces identical results) to perform this test.

1. If the year is not divisible by four, it **is not** a leap year.
2. Otherwise, if the year is not divisible by 100, it **is** a leap year.
3. Otherwise, if the year is not divisible by 400, it **is not** a leap year.
4. Otherwise the year **is** a leap year.

2.2 Days in a Month

The number of days in each month is as follows.

- Months 1, 3, 5, 7, 8, 10 and 12 (January, March, May, July, August, October, December) each have 31 days.
- Months 4, 6, 9 and 11 (April, June, September, November) each have 30 days.
- Month 2 (February) has 29 days in a leap year and 28 days in a non-leap year.

2.3 Computing a Day Index

The `day_index` function takes a year/month/day and returns the "day index" of that date. The day index of a particular date is the number of days from the beginning of the year to that date. For example, the day index of January 1st (in any year) is 1. The day index of December 31st is either 365 (in a non-leap year) or 366 (in a leap year).

To compute the day index for a particular year/month/day combination, you can use the following algorithm.

- Add up the number of days in each month **before** (but not including) the provided month.
- Add the day number.

For example, the day index for March 15th, 2024 (2024-3-15) would be computed as

$$\underbrace{31}_{\text{Jan.}} + \underbrace{29}_{\text{Feb.}} + 15 = 75.$$

(Note that 2024 is a leap year, so February 2024 will have 29 days)

2.4 Chronological Ordering of Dates

Consider two dates $y_1/m_1/d_1$ (Date A) and $y_2/m_2/d_2$ (Date B). To test whether Date A is chronologically earlier or later than Date B, you can use the following algorithm.

- If $y_1 < y_2$, then Date A is earlier than Date B. If $y_1 > y_2$, then Date A is later than Date B.
- Otherwise, if $m_1 < m_2$, then Date A is earlier than Date B. If $m_1 > m_2$, then Date A is later than Date B.
- Otherwise, if $d_1 < d_2$, then Date A is earlier than Date B. If $d_1 > d_2$, then Date A is later than Date B.
- Otherwise, the two dates are equal.

Best Practices for Code Style

Your submissions **must** observe the following guidelines. Submissions which do not meet the guidelines will have marks deducted, even if they function correctly (although some of the guidelines, such as those relating to uninitialized memory, might also affect the ability of the program to work correctly on a reliable basis, which could result in a deduction for both style and function).

- Submitted files must be syntactically correct and named according to the assignment specification.
- Every source file must contain a comment (at or near the top of the file) with your name, student number, the date the code was written and a brief description of the program.
- The `goto` statement is not permitted in any assignment submissions. Instead, your code should use structured control flow elements, like `if`, `for` and `while`.
- Global variables (data variables created outside of the scope of a function) are not permitted, except when explicitly allowed by the assignment. For this assignment, no global variables are permitted, except for `const` variables if needed.
- Every function with a non-`void` return type must return a value.
- Uninitialized variables may not be used before being assigned a value.

3 Evaluation

Your code must compile and run correctly in the CSC 111 virtual lab environment using the compile and run commands given above. If your code does not compile **as submitted** (with no modifications whatsoever), you will receive a mark of zero. Note that submitting an incorrectly-named file will result in the compile command above failing (and the submission receiving a mark of zero).

This assignment is worth 4% of your final grade and will be marked out of 21.

Second Chance Submission

After the initial due date (Sunday, Oct. 8th, 2023) has passed, your code will be automatically tested using a variety of inputs (which will not necessarily match the example inputs shown in the previous sections). If your programs do not give correct output on some of the tested inputs, you will be allowed to fix your program and resubmit it until Wednesday, Oct. 11th, 2023. The grade for the assignment will be computed to be the average of the grade for both submissions (so you can recover some, but not all, of your marks by fixing and resubmitting your code). If you do not resubmit your code, your initial submission will be used in place of the resubmission.

If you do not submit an initial version of your code before the due date, you are **not** permitted to resubmit your code and will receive a mark of zero.

Submission Instructions

All submissions for this assignment will be accepted electronically. Submit your `date_functions.c` file through the Assignment 3 entry on BrightSpace. You are permitted to delete and resubmit your assignment as many times as you want before the due date, but no submissions will be accepted after the due date has passed. For the second chance submission described above, a separate assignment entry will be created after the initial due date.

Ensure that each file you submit contains a comment with your name and student number, and that each file is named correctly (as described in this document). If you do not name your files correctly, or if you do not submit them electronically, it will not be possible to mark your submission and you will receive a mark of zero.

To verify that you submitted the correct file, you can download your submission from BrightSpace after submitting and test that it works correctly in the virtual lab environment. It will be assumed that all submissions have been tested this way, and therefore that there is no reasonable chance that an incorrect file was submitted accidentally, so no exceptions will be made to the due date as a result of apparently incorrect versions being submitted.