CS-102 Design Principles: Project 2

Due date: Monday, February 6 at 11:00pm

In this project, you will make a program that prints a calendar for a given month and year. Please start by accepting the assignment at:

`https://classroom.github.com/a/Je6T4mz9`

which gives you a GitHub repository for your project.

How to submit:

- ☐ add/commit/push your final code to your GitHub repository, this is the code that will be graded.
- ☐ on Gradescope, click on the assignment for this project, and submit your GitHub repository. (Select *master* as your branch.)

Calendar

We see it every day: it's on our phones, it's on our Google accounts, and (for some still) it's on our walls. The monthly calendar is ubiquitous, and in this project you will make one! Given a month and year from the user, your goal is to output a carefully formatted, and correct, calendar for that month and year. Here is the problem:

Name of file: `calendar.cpp`
Input: An integer month m where $1 \leq m \leq 12$, and an integer year y where $1900 \leq y \leq 65535$.
Output: A calendar for the m-th month of year y.

Assume the user always gives correct input: **no error checking is required.**

Here is one example run for February 2021:

```
Enter the month: 2
Enter the year: 2021
February 2021
Su  M  T  W Th  F Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28
```

Here is another example run for May 1940:

```
Enter the month: 5
Enter the year: 1940
May 1940
Su  M  T  W Th  F Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

So, how will you do it?

Anchor day. You will want to pick an "anchor day": a particular day in history where you know the day of the week, and can use it to compute the day of the week on which to start the given month. Given the problem description, January 1, 1900 is a worthy choice. To help you determine which day of the week this is, feel free to search online.

Leap years. Normally, the month of February has 28 days, and the year has 365 days. However, during a leap year, the month of February has 29 days, and the entire year has 366 days.

You need to know how to calculate whether a given year is a leap year. See the method here: `https://www.wikihow.com/Calculate-Leap-Years`

Consider how this will affect your display, and your calculations.

Spacing. The calendar is carefully spaced to be aesthetically pleasing. There are no spaces at the end of lines, and every line has a line ending. Here is a display of the same May 1940 calendar as above, but with all spaces replaced with '`.`' and all line endings replaced with '`\`'. Notice that columns are right-aligned. Excepting the first column and first day, single-digit dates are preceded by exactly 2 spaces, and double-digit dates are preceded by 1 space.

```
May.1940\
Su..M..T..W.Th..F.Sa\
..........1..2..3..4\
.5..6..7..8..9.10.11\
12.13.14.15.16.17.18\
19.20.21.22.23.24.25\
26.27.28.29.30.31\
```

Suggestions for how to proceed with development:

- ☐ Start by ensuring you can accept input from the user and make the calendar's title (e.g., "February 2021")
- ☐ Be able to print out the calendar, assuming the month starts on a Sunday. Start by printing the correct days on the correct lines, then work on spacing. You want this code to be generic enough to make it easy to extend to start any day of the week. The first 4 tests in the autograder check for printing the calendar for months starting on a Sunday.
- ☐ Adjust your code to be able to print a calendar starting on any day of the week.
- ☐ Incorporate leap year calculations to ensure that the user's month/year starts on the correct day.

Grading

Your grade will come from:

75%	correctness + passing the tests
25%	style

There are 15 tests for this project. Projects passing zero tests will receive at most 70% of the project grade (assuming perfect style). Passing tests increases the grade; however, this assumes that your code isn't written to circumvent testing (for example, by "hard coding" answers to the tests), which is a violation of the honor code.

The elements of style that we will assess on Project 2 are:

- ☐ at least 3 well-designed helper functions, aside from `main`, with reasonable names in proper naming style: using `lower_case` naming convention
- ☐ proper indentation
- ☐ correct placement of curly braces and parentheses
- ☐ correct spacing around operators
- ☐ reasonable variable names in proper naming style: using `lower_case` naming in local scope, and `UPPER_CASE` naming for global constants
- ☐ all file and function header comments are present, in the required format, and filled out correctly