Project 1 - Calendar Generator

Due Feb 28 by 11:59pm **Points** 30 **Submitting** a file upload
File Types h and cpp **Available** until Mar 3 at 12:01am

This assignment was locked Mar 3 at 12:01am.

ST. CHARLES COMMUNITY COLLEGE

CPT-182 - Programming in C++

Programming Project - Calendar Generator (30 Points)

(Number in Question Bank: Project **1.1**)

Project Overview

In this project, you are going to write a C++ program that generates a calendar and show it in the console.

Program Input

Your program asks the user to enter **3** input values (see below) via keyboard.

- **Year**. The user should enter a **4**-digit year (e.g., **1997**, **2023**), which should be stored as an `unsigned int`.
- **Month**. The user should enter a **1**- or **2**-digit month (between **1** and **12**). For example, "**5**" means May and "**11**" means November.
- **Day-of-week** of the first day of the month. The user should enter the day-of-week of the first day of "that year" (input year) "that month" (input month). The user should enter a **3**-character string as a day-of-week. For example, **"Mon"** means Monday and **"Thu"** means Thursday. This input value is **not** ~~case-sensitive~~, which means that **"Mon"**, **"MON"**, and **"mon"** are all valid.

In this project, you can assume that the user will always enter the **correct day-of-week** of the first day of the input year and month. You do **not** need to worry about ~~incorrect input values~~.

Program Output

The output of the program is the calendar of the user input year and month. Please see the examples below to better understand the expected output format.

- If we take "October 2019" as example, the expected output format in the console should be:

Console	October 2019 Sun Mon Tue Wed Thu Fri Sat 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

- The calendar should be printed in an exact **27**-character wide window.

```
      October 2019
Sun Mon Tue Wed Thu Fri Sat
              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
<----- 27 characters ----->
```

- First row contains the full month name, a space, and the **4**-digit year. This row should be center-aligned. You need to figure out **how many space characters** need to be printed at the beginning of the line to make sure it is center aligned.

▼ How many spaces needed here?

```
     October 2019
Sun Mon Tue Wed Thu Fri Sat
              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
```

▼ How many spaces needed here?

```
      June 2019
Sun Mon Tue Wed Thu Fri Sat
                          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
```

- Second row is fixed, which includes the **3**-character days-of-week separated by space.

```
      October 2019
Sun Mon Tue Wed Thu Fri Sat
              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
```

- Each column is **3**-character wide and right-aligned. Therefore, you need to **add spaces** when necessary. Also, columns are separated by space.

```
   October 2019
Sun Mon Tue Wed Thu Fri Sat
              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
```

Leap Years

If the user input month is **February**, then your program needs to consider the **leap year** issue, in order to determine whether the February has **28** days or **29** days.

The **pseudocode algorithm** to determine whether a year is a leap year is given below.

```
if year is not divisible by 4 then (it is a common year)
else if year is not divisible by 100 then (it is a leap year)
else if year is not divisible by 400 then (it is a common year)
else (it is a leap year)
```

Other Development Notes

- You **must** use loops to print the dates in the calendar. You **cannot** use ~~exhaustive method~~ to list all the possibilities in your code.
- Please note that program users **cannot** see or understand your code. Therefore, you need to give sufficient **screen prompts** to let the users clearly know what to input in each step.

Project Submission and Grading (Please Read)

- Please upload all your **.h** (if any) and **.cpp** files (**not** the ~~entire Microsoft Visual Studio project folder~~) on Canvas.
- Before the project deadline, you can submit your work unlimited times. However, only your latest submission will be graded.
- At least **20**% of your code should be **comments**. All variable, function (if any), and class (if any) names should "make good sense". You should let the grader put **least effort** to understand your code. Grader will **take off points**, even if your program passes all test cases, if he/she has to put extra **unnecessary** effort to understand your code.
- Please **save a backup copy** of all your work in your computer hard drive.

- Your program will be graded (tested) using multiple different sets of valid input data to check whether it can generate the expected (correct) output. As long as the input values are valid, your program should generate correct output. In other words, your program should work for **any** valid input data, **not** just the sample input data provided in the project instructions.
- In this class, you can assume that all input data are always **valid** and **have correct format**. You do **not** need to deal with ~~invalid input~~ or ~~error handling~~.
- Your work will be graded after the project deadline. All students will receive their project grades at (almost) the same time.