Project #10 (50 points, 10% bonus for exact match): Classes

Submit Your Solution Using Canvas by 10 PM on Thursday, 4/22/21

(late submissions will be accepted on 4/22/21 from 10PM to Midnight)

Obtaining Project 10 Input Order and Output:

Your program's output should look like that produced by the provided sample solution. You can run the sample solution **Project_10_solution** by typing the following at a command prompt in a terminal window**:**

```
/home/work/cpe211/Executables/Project_10/Project_10_solution
```

The comparison script can be run by using the following command

```
/home/work/cpe211data/Project_10/CompareSolution.bash   Project_10.cpp
```

Project 10 Restrictions

Any C++ technique covered in Chapters 1 through 12 is allowed except for global variables ***You are not allowed to use any global variables.*** If necessary, global constants may be used.

Project 10 Description

For this project a Date class (specified in the file **Project_10.h**) will be constructed and all function definitions will be written in the file Project_10.cpp. A **makefile** is provided to compile this project. The file **Project_10_main.cpp** contains the main function that is used to run the program. On Canvas, download the files Project_10.h, Project_10_main.cpp and Makefile. Save these files in your Project 10 directory. **Do not modify these files.**

Another file is provided as well. It **is Project_10.cpp. Download this and all the code you write will be in this file.** This file contains all the function definitions for the methods for the class specification provided in the header file Project_10.h

This project is similar to the Time class described in the classes power point slides and goine over in class

Note: do not use any variable/parameter in your functions or program that are named month, day or year – these are used as private members of the class and you want to make sure that you know which variable or parameter you are talking about.

For this project the following functions are written:

1) For class constructors:
    a. A default constructor called Date() – default date is 1/1/1900
    b. A parameterized constructor Date(int month, int day, int year) – initialize a Date object to the values provided when it is declared(i.e. Date myDate(1,1,1900);)
    c. A parameterized constructor Date(int month,int year) – initialize a Date object with the two values provided and sets the day value to 1.
2) For the class Transformers
    a. A parameterized function SetDate(int,int,int) – this sets the date in a Date object (instance/variable of the class Type Date) to the date specified
    b. A function to increment the date by 1 day

3) For the class observers –two functions for writing out the date – one format is all numbers the other requires the use of the month name and three functions for comparing two dates.
    a. WriteNumberFormat() – writes out the date in format MM/DD/YYYY
    b. WriteNameFormat() – writes out the date in format MonthName day, year
    c. SameDate(Date) – returns true if the class object invoking the function is the same date of the class object passed in as an argument and returns false otherwise
    d. BeforeDate(Date) – returns true if the class object invoking the function is before the date of the class object passed in as an argument and returns false otherwise
    e. AfterDate(Date) – returns true if the class object invoking the function is after the date of the class object passed in as an argument and false otherwise

After downloading all 4 files (Project_10.h, Makefile, Project_10_main.cpp and Project_10.cpp), start your work by making empty function stubs in Project_10.cpp. The function headings are determined from the public functions listed in Project_10.h. Once you have all the function stubs written, you should be able to run the Makefile (type make at the command prompt) to generate the program executable Project_10. You can run the program, but not much happens.

Now add code to each of the functions described above and after completing each function, run the makefile to make sure the program still compiles. If it does not compile fix the errors until it does compile. I recommend writing the two constructors and WriteNumberFormat functions first. Then the rest of the functions can be written in any order.

Project 10 Function Contents

1) The **constructor functions** set the private members of month, day and year to the default values or the values specified. The order for the 3 parameter Parameterized constructor is Month, Day and Year. The order for the 2 parameter Parameterized constructor is Month, Year and the day is set to 1.

2) The **SetDate** function takes in a date – order is Month, Day and Year – and stores those values in the appropriate private members

3) The **IncrementDate** function increments the date of the object by one day. This function will have to make a test for when a new month is moved into and when a new year is moved into because of the increment. Using a switch statement based on the month of the current date simplifies the work necessary for cases where a new month or year is entered due to the increment of 1 day. **Do not worry about leap years – assume February always has 28 days.** Lots of ways to perform the increment action

4) The **WriteNumberFormat** function takes the values of the private members month, day and year and outputs them in the form **mm/dd/yyyy. If needed, Leading 0's are required.**

5) The **WriteNameFormat** function uses the values of the private members month, day and year and outputs them in **MonthName day, year** format. To get the month name, use a switch statement using month as the switch expression and the enumerators provided in the header file as the case labels.

6) The **SameDate** function compares the private members month, day and year of the object that invokes the function with the month, day and year members of the object supplied as the argument in the function call. Boolean true is returned if the dates are identical and false is returned otherwise.

7) The **BeforeDate** function compares the private members month, day and year of the object that invokes the function with the month, day and year members of the object supplied as the argument in the function call. Boolean true is returned if the object that invokes the function has a date that is before the date of the object supplied as the argument in the function call. False is returned otherwise.

8) The **AfterDate** function compares the private members month, day and year of the object that invokes the function with the month, day and year members of the object supplied as the argument in the function call. Boolean true is returned if the object that invokes the function has a date that is after the date of the object supplied as the argument in the function call. False is returned otherwise.

When completed, the program output shall match that of the sample solution. There are no input files for this program. All operations controlling the output are run from Project_10_main.cpp.

Output that does not match that of the sample solution will lose points.

All of these functions are relatively short (less than 10 lines) except for the WriteNameFormat and IncrementDate functions. The WriteNameFormat function has several lines due to the 12 case label switch statement – unless you want to create an array and store the month names in an array and then access the correct array element for the name – just remember the month numbers are 1 through 12 and array indexing starts at 0. There are also ways to create these functions with fewer lines with proper logic and compact code writing. A switch or if then else if statements are more verbose, but straightforward to program.

Submit your Project_10.cpp file only on Canvas.

Project 10 Helpful Information

- To output days or months less than 10 with a leading 0, use **`cout << setfill('0');`** and then output the month or day in a field width of 2 right justified. After printing out the date, use **`cout << setfill(' ');`** to set the fill character back to a space
- After all stubs have been created and the Makefile compiles the program without errors, write one function and then run the Makefile by typing make on the command line. Fix all errors in the function before moving on to the next function.
- **There is very little in the way of output statements in the functions. Matter of fact, only the two write functions have output statements in them. All other output is handled in main which is provided and should not be changed.**
- **Do not modify the Makefile, Project_10.h or the Project_10_main.cpp. The only file that you should be changing is your Project_10.cpp file**