CSCE 240 – Programming Assignment Two

Due: 11:59pm on Friday, September 13th

Program Purpose

Write, test, and use the five functions listed below to output the number of days between two input dates.

Functions

Function 1 – LeapYear

Write a *LeapYear* function that takes a integer argument and returns true if the value is a leap year and false otherwise.

If a year is evenly divisible by 400, it's a leap year. ELSE if it's evenly divisible by 100, it's not a leap year. ELSE if it's evenly divisible by 4, it's a leap year. ELSE it's not a leap year. For example 2000, 2020, and 2024 are leap years. 2023, 2025, and 2100 are not leap years.

Function 2 – LastDayOfMonth

Write a *LastDayOfMonth* function that takes a integer argument for the month and an integer argument for the year. If the month is not 2 (February), the year is not a required argument (should have a default argument 0). The function will return the number of days in the month. For example, LastDayOfMonth(1) should return 31, and LastDayOfMonth(4) should return 30.

If the month is invalid, the function should return 0. For example, LastDayOfMonth(13) should return 0.

If the month argument is 2 (February), the function must be sent a positive integer for the year in order to return the last day of the month. If the year agument is not a positive integer, the function should return 0. For example LastDayOfMonth(2) should return 0. LastDayOfMonth(2, 2023) should return 28. LastDayOfMonth(2, 2024) should return 29.

Function 3 – ValidDate

Write a *ValidDate* function that takes a integer argument for the month, an integer argument for the day, and an integer argument for the year. The function will return true if the values are a valid date, false otherwise.

For example ValidDate(1, 5, 2023) should return true.
ValidDate(7, 50, 2023) should return false.

If the year is not a positive integer, then the function should return falsen

Function 4 – NextDate

Write a *NextDate* function that takes a integer variable argument for the month, an integer variable argument for the day, and an integer variable argument for the year. If the values of the argument are a valid date, the function will update the variables sent to the function to the next calendar date.

For example, if the function is sent variables containing 1, 31, and 2023, the function update those variables so that they hold 2, 1, and 2023 when the function completes.

If the values of the arguments are not a valid date, the variable arguments will remain unchanged. For example, if the function is sent variables containing 2, 29, and 2025, the function should leave the variables unchanged.

Function 5 – PreviousDate

Write a *PreviousDate* function that takes a integer variable argument for the month, an integer variable argument for the day, and an integer variable argument for the year. If the values of the argument are a valid date, the function will update the variables sent to the function to the previous calendar date. If the values of the arguments are not a valid date, the variable arguments will remain unchanged.

For example, if the function is sent variables containing 1, 1, and 2023, the function should update those variables so that they hold 12, 31, and 2022 when the function completes.

program2.cc

Write a program2.cc source file with a main function that accepts two input dates (using cin) in the format: int char int char int. The program should output the number of days between the two dates in the format:

firstdate is # days *before/after seconddate*

If either date is invalid, the program should output

date is not a valid date

and exit.

Example input/output pairs

Input: 12/30/2022 1/2/2023
Output: 12/30/2022 is 3 days before 1/2/2023

Input: 3/7/2023 2/20/2023
Output: 3/7/2023 is 15 days after 2/20/2023

Input: 5/8/2000 5/8/2000
Output: 5/8/2000 is 0 days before 5/8/2000

Input: 2/6/2002 2/29/2002
Output: 2/29/2002 is not a valid date

Input: 1/3/2025 4/31/2018
Output: 4/31/2018 is not a valid date

Specifications

- All output should be directed to the standard output device using cout.
- All input should be accepted from the standard input device using cin.
- Do not prompt for input.
- The prototypes for the *LeapYear*, *LastDayOfMonth*, *ValidDate*, *NextDate*, and *PreviousDate* functions must be included in *program2functions.h*

- The *LeapYear*, *LastDayOfMonth*, *ValidDate*, *NextDate*, and *PreviousDate* functions must be implemented in *program2functions.cc*
- Your main function must be implemented in *program2.cc*
- You will submit a zip file (only a zip file will be accepted) containing *program2functions.h*, *program2functions.cc*, and *program2.cc* to the assignment in Blackboard.
- Programs must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).
- Be sure to review the program expectations section of the course syllabus.

Testing

Initial tests have been included for the functions in the attached files *testLeapYear.cc*, *testLastDayOfMonth.cc*, *testValidDate.cc*, *testNextDate.cc*, and *testPreviousDate.cc*. You should ensure that your functions pass the included tests, and you are encouraged to create more rigorous tests. Your functions will be graded using tests similar to the included tests, with different values.

Text files containing sample input and the corresponding expected output for *program2.cc* are also attached to the program assignment. A makefile has been included to run your program with the sample input and compare the results to the expected output. In order to use the makefile, ensure that your *program2functions.h*, *program2functions.cc*, *program2.cc*, and all of the files attached to the assignment (*checkit.cc*, *correct-test1.txt*, *correct-test2.txt*, *correct-test3.txt*, *correct-test4.txt*, *test1-input.txt*, *test2-input.tx*t, *test3-input.txt*, *test4-input.txt*) are in the same directory. Your program will be graded using this same method with different input/output file pairs.

The commands to run the tests are given below:

```
make testLeapYear
make testLastDayOfMonth
make testValidDate
make testNextDate
make testPreviousDate
make testprogram2a
make testprogram2b
make testprogram2c
make testprogram2d
```

Note: Differences in capitalization or spacing (including extra whitespace at the end of the output) and prompts for input will cause the tests to fail. End your last output statement with endl. The tests will display your output up to the first character that doesn't match the expected output. You can view your full output in the *student-test#.txt* file and compare it to the corresponding expected output in the *correct-test#.txt* file.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compilation of *program2functions.cc*: 1 point
Clean compile and link of *program2.cc*: 1 point
LeapYear passes instructor tests: 1 point
LastDayOfMonth passes instructor tests 1: 1 point
ValidDate passes instructor tests: 1 point
NextDate passes instructor tests: 1 point
PreviousDate passes instructor tests: 1 point
program2 runs correctly with instructor input: 1 point

The penalty for late assignment submissions is 10% per day up to three days after the assignment due date. No assignment submissions will be accepted more that 3 days after the due date.