Programming Project: Isolation Self-Assessment Tool

CPSC 1020

School of Computing
COMPUTING
Clemson University

A university implements the following measures to prevent the spread of COVID-19:

- **Case 1**: If a student tests positive, they must isolate for **7 days**
- **Case 2**: If a student tests negative and they were not exposed to someone who tests positive, they do not have to isolate
- If a student tests negative but they were exposed to someone who tests positive, they must:
    - **Case 3**: isolate for **3 days** if they were fully vaccinated at the time of exposure
    - **Case 4**: isolate for **12 days** if they were not fully vaccinated at the time of exposure
- A student is considered fully vaccinated after two weeks of receiving their second vaccination dose

Write a program that provides students with a self-assessment tool they can use to determine if and for how long they must isolate. Your program should ask the user a series of questions.

Based on the information provided by the user, your program should record the date they tested positive, the date they were exposed, and the date they received their second dose. Only record information that is necessary. For example, if a student tests positive, your program should tell them to isolate for 5 days regardless of their vaccination status or whether not they were exposed.

Additional specifications:

- For simplicity, your class should be restricted to record dates between 01/01/2023 and 12/31/2024.
- Use in-class initialization for all member variables. Use 01/01/2023 as the default date.
- Use the constructor `Date(d: int, m: int, year: int)` with member list initialization to allow creating an instance of `Date` with user-provided values. Your constructor must perform input validation. If the user provides an invalid value (e.g., 13 for month) or a value that would lead to an invalid date, the constructor should set all variables to their default

values.

- Each setter function should return a Boolean value that should be used for input validation in the main program. If the user provides an invalid value (e.g., 13 for month) or a value that would lead to an invalid date, a setter function should return false and leave the corresponding variable unchanged.
- Use function `showDate` to return a string formatted as “MM/DD/YYYY” (e.g. 02/15/2023)

Classes

Create a class `Date` as specified in the below UML Class Diagram:

Date
- day: int - month: int - year: int
+ Date() + Date(d: int, m: int, y: int) + setDay(d: int): bool + setMonth(m: int): bool + setYear(y: int): bool + getDay(): int + getMonth(): int + getYear(): int + showDate(): string + addDays(days : int)

Date Comparison

If a student tests negative but was exposed, their isolation length depends on whether they were fully vaccinated. To decide their vaccination status at the time of exposure, you need to compare the date they were exposed to the date they received their second vaccination dose (+ 2 weeks to be considered fully vaccinated).

Create a function `calcDays` that takes as input two instances of `Date`. This function should return the number of days between the two dates recorded in date1 and date2. Use pass by const reference to pass date1 and date2 to the function.

Do NOT use a library to calculate the difference between date1 and date2. Instead, implement this simple algorithm:

1. Calculate number of days between 01/01/2023 and date1 as daysDiff1
2. Calculate number of days between 01/01/2023 and date2 as daysDiff2
3. Calculate number of days between daysDiff1 and daysDiff2.

Use the number returned by `calcDays` to determine a student's vaccination status at the time of exposure.

Client Program Requirements

Depending on the user input, your program should create the following instances from class `Date`: datePositive, dateExposed, and dateSecondDose. Only create an instance if necessary. For example, if a student replies they tested positive, their vaccination status and whether they were exposed is irrelevant.

It is your choice how you format the questions to collect user information. However, all user input to record a date must be validated as explained above.

After your program has collected all necessary information and made all necessary calculations, your program should tell the user if and how long they must isolate together with all data they have entered. Format your output as follows

Example output:

- Case 1 from above

```
Test result: positive
Date tested positive: 02/01/2023
Length of isolation: 7 days
Return to classes: 02/08/2023
```

- Case 2 from above

```
Test result: negative
Exposed to positive case: No
Length of isolation: 0 days
```

- Case 3 from above

```
Test  result:  negative
Exposed to positive case: Yes
Date exposed to positive case: 02/01/2023
Second vaccination dose received: Yes
Date second vaccination dose received: 01/01/2023
Vaccination status at time of exposure: fully vaccinated
Length of isolation: 3 days
Return to classes: 02/04/2023
```

- Case 4 from above

```
Test  result:  negative
Exposed to positive case: Yes
Date exposed to positive case: 02/01/2023
Second vaccination dose received: Yes
Date second vaccination dose received: 01/31/2023
Vaccination status at time of exposure: not fully vaccinated
Length of isolation: 12 days
Return to classes: 02/13/2023

Test  result:  negative
Exposed to positive case: Yes
Date exposed to positive case: 02/01/2023
Second vaccination dose received: No
Vaccination status at time of exposure: not fully vaccinated
Length of isolation: 12 days
Return to classes: 02/13/2023
```

Design and Pseudocode

Write a description of your program in pseudocode before you start coding. Pay particular attention to control flow when designing your program. Carefully think about the sequence in which you collect input from the user and what objects you need to create. Remember that pseudocode is written for humans, not for computers, but should be detailed enough so that it can be translated into code.

SUBMIT YOUR PSEUDOCODE A WEEK BEFORE PROGRAM

Submission Instructions

- Submit your design document with pseudocode via Canvas.
- Submit the following 4 files via Gradescope:
    - main.cpp
    - Date.h
    - Date.cpp
    - calcDays.h
    - calcDays.cpp
    - Makefile
- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.
- Code style (proper indentation, consistency, readability, use of comments, etc.) will be considered when grading your submission.