ENSC 251 Lab Assignment 2

Lab Assignment Overview:

In the ENSC 251 course, you will work on four lab assignments using the skills learned throughout this course, i.e., Object-Oriented Programming (OOP) with C++. Each lab assignment weighs 10 marks. All lab assignments will be carried out and evaluated in pairs (i.e., two students per group) - I hope by now all of you have already found a partner. Some general grading logistics have been posted on our course website: https://canvas.sfu.ca/courses/70216/pages/lab-logistics.

Please make sure your code can compile and run correctly on the lab computer. If your code cannot compile, then your group can get at most 5 marks. If your code can compile, but cannot run, then your group can get at most 6 marks.

You can divide the work roughly based on the listed tasks; the overall marks for major tasks are also listed for your guidance, but not the detailed marks for every single item. Please include a simple document in each lab (include a txt or pdf file in the .zip file for submission) indicating which parts each member finished. There is some negative marking if you don't complete the listed tasks; details are not listed. Basically, the detailed grading scheme (except the overall marks for major tasks) for each lab will NOT be released before your lab grading is done. Think about you are in a real interview, nobody will tell you what the detailed grading schemes are.

Lab Assignment 2:

In the first two lab assignments, you will build a simple calendar system, where one can book, query, and delete appointments. To make it simpler, we make the following assumptions:

1) The dates supported in the calendar system are from May 9, 2022 to Aug 8, 2022, i.e., the Summer term of 2022.
2) No appointment can be made on weekends and holidays.
3) The time format for each day is using the 24-hour format, from 0:00 to 24:00. We only care about the hour and minute field and ignore the second field.
4) An appointment in the calendar system can only be made at multiples of 30 minutes, and it only supports the same day appointment. For example, May 9th 9:00 to 9:30 appointment is valid, May 9th 10:00 to 12:30 is also valid, May 9th 10:10 to 10:30 is invalid because 10:10 is not a multiple of 30 minutes, May 9th 19:00 to May 10th 9:00 is invalid because it crosses two days.

In lab assignment 1, you have already built the *Date* and *TimeRange* classes. Now let's finish building this simple calendar system in lab assignment 2. **Specifically, in lab assignment 2, you will build the following items.**

1) [7 marks] **Implement the *Calendar* class.** The Calendar class should have member variables to denote all the dates available between May 9, 2022 to Aug 8, 2022, which can be implemented using arrays of Date class objects. Please implement the constructor function(s). No specific need of get and set functions for this class. Moreover, implement the following member functions:

   a. [1 mark] **bool book(const Date &date, const TimeRange &time);** This function books an appointment at the given date and time, assuming it is a valid workday,

and the given valid time range of the day has not been booked yet. If the date or time range is invalid, you should print out the error messages and return false. If it cannot be booked for a valid date and time range, you should print out a hint to the user, e.g., it is a holiday, or it is a weekend, or it is a workday but the input time range or part of the input time range has already been booked. Also return false. In addition, print out the available time slots for the input day that haven't been booked yet. If it can be booked, you should update the calendar and print a hint to the user that it has been booked, and return true.

b. [1 mark] **bool query(const Date &date, const TimeRange &time);** This function searches the current calendar and tells the user whether the given date and time range has been booked or partially booked. Again, don't forget about the error checking.

c. [1 mark] **bool cancel(const Date &date, const TimeRange &time);** This function first searches the current calendar. If the given date and time range has been booked or partially booked, delete the corresponding booking and make those time slots available for booking again. Again, don't forget about the error checking.

d. [1 mark] Implement **printFreeDays()** member function to print all days that have some free time slots in the calendar which can be booked. Don't include holidays and weekends.

e. [1 mark] Implement **printFreeTimeSlots(const Date &date)** member function to print all free time slots for the given date in the calendar, which can be booked.

f. [1 mark] **Overload the << operator** to print out all the booked time slots (including dates and their corresponding time ranges) in the calendar.

g. Make sure that you print out user friendly information, not just array indices.

2) Make sure the Calendar class is implemented as abstract data type (ADT).

3) Add calendar.hpp and calendar.cpp and update Makefile. Make sure your Calendar class is declared and defined in calendar.hpp and calendar.cpp, respectively.

4) For each header file, use the #ifndef macro to avoid double include.

5) [2 marks] **Test your simple calendar system in main.cpp:**

a. Your main program should create a Calendar object for the Fall term and print out a menu to the user when it starts running. This menu provides options for a user to choose among: i) print out all booked appointments; ii) print out all days that have free time slots which haven't been booked; iii) print out all free time slots for a given day which haven't been booked; iv) book an appointment; v) query an appointment; vi) cancel an appointment.

b. For options iii) to vi), first let the user input the date and/or time with prompt. Based on user input, create the corresponding Date and TimeRange object, and call the corresponding member function of the Calendar object.

c. When there is an error in the user input, print out the right hint and error message, and prompt the user to try again.

6) [1 mark] Note the above implementation has an issue. It cannot print out the original appointment info, e.g., a single appointment on Jun 1, 10:00-13:00. You only know Jun 1, 10:00-13:00 has been booked, but do not know which appointments are booked for this time range (e.g., there might be two appointments, one for 10:00-11:00 and the other for 11:00-13:00). Please enhance it so that you can print out exactly which appointments are booked.

In addition to the actual coding implementation, you need to provide good commenting, naming, and other good coding styles; all these count in your lab assignment marking.

Note: For grading logistics and remote machine access, please refer to the course website. If you have any questions, please post them on the discussion board.

Assignment Submission:

Your lab assignment 2 will be submitted electronically through Canvas. You will need to submit a single lab2.zip file. Failure to comply with this format will result in a **ZERO** score for this assignment. To zip your files in Linux,

1. Go to your lab2 directory
2. make clean //**make sure you clean your files**
3. cd .. //go one level up
4. zip -r lab2.zip lab //zip all your lab2 files into a single lab2.zip

Submission Deadline:

Your lab assignment 2 is due at **11:59:59pm on Tuesday, Jun 14th, 2022**. You need to meet the deadline: every 10 minutes late for submission, you lose 1 mark; that is, 100 minutes late, you will get zero for this lab.

Lab Demonstration:

You will have to demo your lab assignment 2 to your TA in the following lab sessions that you enrolled on **Monday (Jun 20th, 2022**). Only code from your Canvas submission is allowed in the lab demo. Each student group has around 10 minutes to explain your code to the TA. If you fail to do the demo (without a medical note), or if it is determined that you do not understand the code being evaluated, you will be awarded zero on this lab assignment. Also please show up in the demo day on time (TA will send out your scheduled time), otherwise you will lose 0.5 mark.