CSCE 2004 – Programming Project 7
Final Due Date – May 4, 2017 at 11:59pm

1. Problem Statement:

The primary goal of this programming assignment is to give students experience designing and implementing classes. To do this, students will design the interface for a calendar "Date" class, and produce a "skeleton implementation" of this class, where the methods all print a message with the name of the method. Detailed requirements are listed below:

- Your first task is to decide what pieces of information you want to store in each object. Once you have done this, you need to create appropriate variables and put them in the appropriate place in the Date class definition in the "date.h" file.

- Your second task is to decide what operations to perform on each Date object. Naturally you will want some form of "get" and "set" methods. In addition to these methods, you should invent three other methods that perform useful operations on Date objects. For all methods, you will need to decide what parameters are needed, and what the method will return. Put the function prototypes for all of your methods in the appropriate place in the Date class definition in the "date.h" file.

- Your next task is to create a "skeleton implementation" of all of the methods in the "date.h" class in a new file called "date.cpp". Each method should have comments explaining what the purpose of the method is, and each method could have cout statements that print out the name of the method, and any parameters that are passed into the method.

- Your final task is to create a very basic main program at the bottom of the "date.cpp" file who's only purpose is to create several Date objects and call each of the methods in the Date class several times. Because you only have a skeleton implementation above, your program will not do any real work, but it will print out messages that prove that your methods above are getting called and are receiving parameters correctly.

2. Design:

For this assignment, most of your work will be in designing the Date class described above. You should model your design on other classes we have spoken about in class or on the class website. Your goal here is to come up with sensible names for private variables and public methods, and to decide what parameters all of these methods will require.

3. Implementation:

You are starting this programming project with a blank piece of paper, so you do not need to worry about interfacing with existing code. When implementing your project, you should model your "date.h" and "date.cpp" files on other examples you have seen in class. Remember the easiest way to create a skeleton implementation is to copy/paste the function prototypes from "date.h" into "date.cpp" and add the code to print messages. Since you are not actually creating a full implementation of each method, be sure to include a descriptive comment to explain what each method is expected to do.

As always, it is very important to make changes incrementally one feature at a time, writing comments, adding code, compiling, and debugging. This way, you always have a program that "does something" even if it is not complete.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

5. Documentation:

When you have completed your C++ program, write a short report using the "Programming Project Report Template" describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this project report in a separate document to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and your C++ program (a single cpp or txt file). Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

10% off if less than 1 day late,
20% off if less than 2 days late,
30% off if less than 3 days late,
no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations