CSIT 840 Object-Oriented Programming in C++	Devvie Schneider Kent Assignment #3 (Due Date per Website)

Preface

This assignment is associated with ***Modules 7, 8, and 9,*** and the textbook material referenced in those modules.

The ancient programmer, Genghis Khent, had a social life was typical for a computer programmer; non-existent. He wanted a date. Since a C++ class can represent anything in the real world, he ordered his favorite source of free C++ programming labor, his programming students, to write a class to represent a date. Unfortunately, the students misunderstood Genghis Khent. Their Date class represented a calendar date.

Frustrated and angry, Genghis Khent decided to punish his students by overloading them. He did so using overloaded operators.

This story has been passed on for centuries in Genghis Khent's family. You have to write the same program Genghis Khent's students did.

Help

While you may feel you're beyond help, help is available, specifically Modules 7-9 (*Assignment Operator and Copy Constructor*, *Pointers as Class Member Variables*, and *Operator Overloading*) and the sections of the textbook referenced in those modules. Also, when you do run into problems or questions, please make use of the Assignment #3 discussion forum. Remember, you can cash in discussion participation for credit toward your course grade; this includes any attempts to contribute replies to others. Finally, please don't wait until the last day or two to start posting your questions! You have more than three weeks to complete this assignment.

What the Program Does

The program will ask the user to input two dates, by month, day and year. You may assume the two dates are valid, and that the year is no earlier than 1800. The date may be today's date, a past date (though no earlier than 1800) or a future date.

The following sections show sample runs of input and output, and provide you with the code for the driver file. These sample runs and the driver file code show the program flow. Here is a written description of the program flow:

- The input of the two dates is done using the overloaded extraction operator, >>.
- The two dates then are output using the overloaded insertion operator, <<.
- The two dates then are compared using the overloaded relational operators == and >. If the two dates are equal, the output will be that the two dates are the same. Otherwise, the program will output which date is the more recent and, using the overloaded subtraction operator, by how much (using the month/day/year format, so 6/8/59 means 59 years, 6 months, 8 days). As discussed below, I won't be that picky about the accuracy of the subtraction operator; "close enough for government work" will suffice. Thus, your results may be plus or minus a few days different from the sample runs.

Sample Runs

Sample Run #1 (two input dates are equal):

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 9
Enter day (1 to maximum days in month): 11
Enter year (>= 1800) : 2001
Enter another date no earlier than 1800
Enter month (1 - 12): 9
Enter day (1 to maximum days in month): 11
Enter year (>= 1800) : 2001
Revised date values
Date 1 is 9/11/2001
Date 2 is 9/11/2001
The two input dates are the same
After the assignment date4 = date3 = date2
   Date 3 is 9/11/2001 and Date 4 is 9/11/2001
```

Sample Run #2 (the first input date is the more recent):

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 10
Enter day (1 to maximum days in month): 10
Enter year (>= 1800) : 1980
Enter another date no earlier than 1800
Enter month (1 - 12): 4
Enter day (1 to maximum days in month): 2
Enter year (>= 1800) : 1921
Revised date values
Date 1 is 10/10/1980
Date 2 is 4/2/1921
Date 1 is later in time than Date 2 by 6/8/59
After the assignment date4 = date3 = date2
   Date 3 is 4/2/1921 and Date 4 is 4/2/1921
```

Sample Run #3 (the first input date is the more recent):

```
Initial date values
Date 1 is 1/1/2001
 Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 4
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2000
Enter another date no earlier than 1800
Enter month (1 - 12): 3
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2000
Revised date values
Date 1 is 4/30/2000
Date 2 is 3/30/2000
Date 1 is later in time than Date 2 by 1/0/0
After the assignment date4 = date3 = date2:
   Date 3 is 3/30/2000 and Date 4 is 3/30/2000
```

Sample Run #4 (the second input date is the more recent):

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 12
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2005
Enter another date no earlier than 1800
Enter month (1 - 12): 2
Enter day (1 to maximum days in month): 22
Enter year (>= 1800) : 2007
Revised date values
Date 1 is 12/30/2005
Date 2 is 2/22/2007
Date 2 is later in time than Date 1 by 1/23/1
After the assignment date4 = date3 = date2
  Date 3 is 2/22/2007 and Date 4 is 2/22/2007
```

Sample Run #5 (the second input date is the more recent):

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 12
Enter day (1 to maximum days in month): 31
Enter year (>= 1800) : 2005
Enter another date no earlier than 1800
Enter month (1 - 12): 12
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2007
Revised date values
Date 1 is 12/31/2005
Date 2 is 12/30/2007
Date 2 is later in time than Date 1 by 11/29/1
After the assignment date4 = date3 = date2:
   Date 3 is 12/30/2007 and Date 4 is 12/30/2007
```

Sample Run #6 (the first input date is the more recent)

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 11
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2000
Enter another date no earlier than 1800
Enter month (1 - 12): 10
Enter day (1 to maximum days in month): 31
Enter year (>= 1800) : 2000
Revised date values
Date 1 is 11/30/2000
Date 2 is 10/31/2000
Date 1 is later in time than Date 2 by 0/30/0
After the assignment date4 = date3 = date2:
Date 3 is 10/31/2000 and Date 4 is 10/31/2000
```

Sample Run #7 (the first input date is the more recent)

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 11
Enter day (1 to maximum days in month): 5
Enter year (>= 1800) : 2000
Enter another date no earlier than 1800
Enter month (1 - 12): 12
Enter day (1 to maximum days in month): 5
Enter year (>= 1800) : 2000
Revised date values
Date 1 is 11/5/2000
Date 2 is 12/5/2000
Date 2 is later in time than Date 1 by 1/0/0
After the assignment date4 = date3 = date2:
Date 3 is 12/5/2000 and Date 4 is 12/5/2000
```

Sample Run #8 (the first input date is the more recent)

```
Initial date values
Date 1 is 1/1/2001
Date 2 is 1/1/2001
Enter a date no earlier than 1800
Enter month (1 - 12): 11
Enter day (1 to maximum days in month): 30
Enter year (>= 1800) : 2000
Enter another date no earlier than 1800
Enter month (1 - 12): 12
Enter day (1 to maximum days in month): 29
Enter year (>= 1800) : 1999
Revised date values
Date 1 is 11/30/2000
Date 2 is 12/29/1999
Date 1 is later in time than Date 2 by 11/1/0
After the assignment date4 = date3 = date2:
Date 3 is 12/29/1999 and Date 4 is 12/29/1999
```

Driver File

The complete code for your test driver file is given to you here. In the final version of your program submission ***do not change this given code.*** Of course, as you gradually build your program, implementing a function or two at a time, you may want to comment out various parts of this test driver until you have implemented all the required member functions (described below).

```
//test.cpp
#include "date.h"
#include <iostream>
using namespace std;
int main()
{
  Date date1;
  Date date2 = date1; //copy constructor called

  cout << "Initial date values\n";
  cout << "Date 1 is ";
  cout << date1 << endl;
  cout << "Date 2 is ";
  cout << date2 << endl;
  cout << "Enter a date no earlier than 1800\n";
  cin >> date1;
  cout << "Enter another date no earlier than 1800\n";
  cin >> date2;
  cout << "Revised date values\n";
  cout << "Date 1 is ";
  cout << date1 << endl;
  cout << "Date 2 is ";
  cout << date2 << endl;

  if (date1 == date2)
    cout << "The two input dates are the same\n";
  else if (date1 > date2)
  {
    cout << "Date 1 is later in time than Date 2 by ";
    Date temp = date1 - date2;
    cout << temp << endl;
  }
  else
  {
    cout << "Date 2 is later in time than Date 1 by ";
    Date temp = date2 - date1;
    cout << temp << endl;
  }

  Date date3, date4;
  date4 = date3 = date2; //overloaded assignment operator called
  cout << "After the assignment date4 = date3 = date2\n";
  cout << " Date 3 is " << date3 << " and Date 4 is " << date4 << endl;
  return 0;
}
```

Date Class

Now that you are an expert on OOP, you will use a class, naturally enough called *Date*, to represent a date. This class will have variables to hold the month, day and year of a particular date. As with your first assignment, the class will be written using two files, a class declaration file (.h) and a class implementation file (.cpp). Of course, the program also will require a third file, the driver file (given above) containing *main*, to run your program.

Date Class - Member Variables

The *Date* class has three private member variables and no others:

(1) *month*, which represents the month portion of a date ;
(2) *day*, which represents the day portion of a date; and
(3) *year*, which represents the year portion of a date.

For example, if the date was 12/31/2002, *month* would be 12, *day* would be 31 and *year* would be 2002. All three variables are of the integer data type.

Date Class - Constructors/Destructor

The *Date* class should have two constructors:

1. No arguments - initializes the *month*, *day* and *year* variables to a date of your choosing
2. Copy constructor - you may not need it in this implementation, but I want to check that you can write one.

You probably will not find a need for a destructor though you can implement one if you want to.

Date Class - Overloaded Operators

You need to overload the following operators as ***member*** functions:

1. The overloaded equality.(==) operator returns true if the *Date* object calling the == operator represents the same date as *the Date* object that is argument, false otherwise.
2. The overloaded greater than (>) operator returns true if the *Date* object calling the > operator is later in time than the *Date* object that is argument, false otherwise. For example, July 7, 2003 is later in time and therefore larger than December 31, 2002.
3. The overloaded subtraction (-) operator returns a *Date* object representing difference between the *Date* object calling the - operator and the *Date* object that is argument. ***I won't be that picky about the accuracy of the subtraction operator; "close enough for government work" will suffice. Thus, your results may be plus or minus a few days different from the sample runs.***

4. The overloaded assignment operator. You may not need it in this implementation, but I want to check that you can write one.

Note: Although the book gives examples of overloading the arithmetic and relational operators as friend functions (and admittedly, there are some real advantages to this approach), I am requiring you to overload them as member functions to demonstrate that you understand which operand instance is implicitly passed though the "this" pointer.

You also need to overload the following operators as ***friend*** functions:

- Extraction (>>) - enables use of *cin* >> on a *Date* object.
- Insertion (<<) - enables use of *cout* << on a *Date* object.

Date Class - Member Functions

You may write additional member functions for the *Date* class, so long as none duplicate what any of the overloaded operators do. I want you to use the overloaded operators in your program so I don't want you to write any member functions that enable you to avoid using the overloaded operators. Therefore, for example, you can't write a print member function because that would avoid use of the overloaded insertion stream operator.

Summary of Restrictions:

- Use the test driver code that is given above. You may not change it, except to comment out code as you build your project in pieces or to comment out code that calls a member function which you are not able to implement.
- You will need to declare the 3 private member variables listed above, but do not include any additional member variables. Also, these 3 member variables must be declared with *private* access.
- Neither of the overloaded relational operators (that is, equality and greater-than) nor the overloaded subtraction operator should be declared as "friend" functions for the reason specified in the "***Note***" above. Rather, they must be "member" functions.
- Do not include any member functions which allow you to avoid using any of the required four overloaded operator member functions.
- You will need to write the two required constructors and four required overloaded operator member functions listed above.

Additional Requirements

Your project must have 3 files, no more and no less; *date.h*, the declaration file for the Date class, *date.cpp*, the implementation file for the Date class, and *test.cpp*, the "driver" file which tests the class. You cannot put all your code in one file, or create additional files. You also cannot change the driver file code I have given you above.

Please be sure to include a comment at top of all three of your files which includes your name (at the very least).