Self-Evaluation Report for Programming Assignments

Section I: Basic Information

1. **Programming assignment #: 2**
2. **Name of the author: Esther Choi**
3. **Name of the peer reviewer, if any: Matt Robles, Paul Choi**
4. **Due date of the assignment: March 1, 2017**
5. **Date when the assignment was finished: March 1, 2017**
6. **Number of hours spent in programming: roughly 4.5 hours**

Section II: Integrity Review

Integrity rules for regular programming assignments

- **Peer discussion**: Peer discussion of code shown on a screen or board is acceptable for explanation of ideas and for debugging purpose. Such discussion may help to cultivate an open learning environment in the class, but you should carefully read the guidelines below to avoid any dishonest behavior and never step over the guidelines explicitly described in the following.
- **Never use any code (i.e. C++ statements, segments of a program or an entire program) written by others (except for examples in our textbooks or reading)**: Any copy-and-paste of code from other people's programs or from websites is viewed as cheating and you will get 0 points for the assignment.
- **Never circulate your code to others**: You should never pass around your code (electronically or on paper) to others except for the TA and the instructor. Violating this rule is viewed as cheating in the class and the provider will receive 0 points for the assignment.
- **Never provide false or exaggerated results of test cases**: You need to report results of test cases in the self-evaluation report together with all your source code files for each assignment**.** Providing false or exaggerated results of test cases in the report is viewed as cheating and you will receive 0 points for the assignment.
- **Demonstrate the credibility of your authorship of the work**: When you submit your code as your own work for points, you should make sure that you are able to explain your code and reconstruct your code from scratch without any outside help when requested. If you are not able to do that on your own when requested, you will get 0 points for the assignment and there will be an investigation.
- **Consequence of cheating in the class**: Cheatings end in 0 points for the assignments followed by discipline actions described in the student handbook.

1. **Have you ever received any code written by others? No**
2. **Have you ever passed any code you wrote to others? No**
3. **Have you ever used any code written by others? No**

Section III: Test cases and peer review

Note: To get all the points, you should have a peer reviewer watch the behavior of your program before you submit the work. You should prepare your own test cases and have your reviewer see the results when you run your program over the test cases. Optionally, you may also have the reviewer run your program through the reviewer's own test cases to see whether your program works correctly.

1. **Compile and run your code using Visual C++ 2013 as the testing environment and describe the test cases used and the results you and the reviewer have observed**:

SetRandomDate()		
Input	**Testing For**	**Results**
	Testing rand(), round 1	2/11/119 1/18/9727 7/12/2752 1/29/2778 1/17/1527 10/5/1271 8/9/4110 1/30/5361 2/3/1754 4/22/9203
	Testing rand(), round 2	12/8/2675 4/14/3243 2/19/9876 12/19/6121 1/4/9440 9/16/6584 8/21/1674 9/26/2903 6/12/2458 10/22/9489
	Testing rand(), round 3	7/5/1621 7/1/2763 9/26/1278 8/10/8054 5/17/2749 2/6/2998 11/30/2661 10/5/3276 12/11/1623 9/30/1970

Relational Operators				
Input (date1, date2)	**Testing For**	**Results**		
11/2/2000, 12/26/1990	When date1 is greater than date2	(date1 > date2)	Yes	True
		(date1 < date2)	No	False
		(date1 == date2)	No	False
		(date1 != date2)	Yes	True
		(date1 >= date2)	Yes	True
		(date1 <= date2)	No	False
12/26/1990, 11/2/2000	When date1 is less than date2	(date1 > date2)	No	False
		(date1 < date2)	Yes	True
		(date1 == date2)	No	False
		(date1 != date2)	Yes	True
		(date1 >= date2)	No	False
		(date1 <= date2)	Yes	True
1/1/2000, 1/1/2000	When date1 and date2 are equivalent	(date1 > date2)	No	False
		(date1 < date2)	No	False
		(date1 == date2)	Yes	True
		(date1 != date2)	No	False
		(date1 >= date2)	Yes	True
		(date1 <= date2)	Yes	True
0/0/0	Incorrect date entry test	Error message displays and program loops until proper entry is received		
Arithmetic Operators				
Input (date1, date2)	**Testing For**	**Results**		
1/1/2000, 5/13/2016	When date1 is greater than date2	(date1 - date2)	-5977	
5/1/2016,	When date1 is less	(date1 - date2)	5965	

1/1/2000	than date2		
1/1/1900, 1/1/1900	Where date1 and date2 are equivalent	(date1 - date2)	0
-9/5/2000	Incorrect date entry test	Error message displays and program loops until proper entry is received	
Input (date1, i)	**Testing For**	**Result in date2**	
1/1/1900, 763	Subtracting positive numDays days from date1	(date1 – numDays)	11/29/1897
1/1/1900, 0	Subtracting non-positive numDays days from date1	(date1 – numDays)	Error message displays and program loops until positive numDays is entered
1/1/1, 9	Going back from year 1 (into year 0 or less)	(date1 – numDays)	Displays an error message, and loads date2 with values from default constructor
1/1/1900, 5000	Adding positive numDays days from date1	(date1 + numDays)	9/10/1913
1/1/1900, -4	Adding non-positive numDays days from date1	(date1 + numDays)	Error message displays and program loops until positive numDays is entered

Console Input / Output	
cin (month, day, year)	**cout**
1, 1, 1896	1/1/1896
0, 1, 1	Error message displays and re-entry is required until proper date entry received

File Input / Output	
Input (month, day, year)	**Results**
1, 1, 1	User-defined text file created and 1 1 1 written to said file
1, -8, 2000	Error message displays and re-entry is required until proper date entry received

2. **Description of bugs or other problems discovered by you or the peer reviewer, if any:**

   This is not really a bug, but in version 2 of my program (uploaded to Canvas), I changed the implementation for the + and – overloaded operators. I realized I could do it without using the for loops if I just use AdvanceDays() and BackDays() instead (see DateType.cpp || lines 598, 643). I was like, *'Duh, Esther. So obvious…'* I then reran menu option A using the test cases above and asked my brother to peer review the changed implementation (this is why I have two peer reviewers listed for this program). I ran it on Monday with Matt from CSCI 106 as my peer reviewer, and then changed my implementation on Wednesday. Since I was pressed for time, I asked Paul to act as my peer reviewer and ran just the changed implementation once more using the same test cases listed above in question 1.

   This is also not really a bug, but I handled the non-positive numDays issue differently from the demo program (see DateType.cpp || lines 570-81, 615-26). The program specifications didn't really specify that we had to do it any specific way, so I just did it the way I thought worked best with the program. I also did it the way the demo program handles it and left it commented out so you know I know how to do it (see DateType.cpp || lines 583-96, 628-41).

   Also, the demo program prints to output file by separating the month, day, and year onto individual lines, but the program specs told us to just separate the three with spaces, so I just followed the specifications. Hope that's okay.

   Aside from that, everything else is according to both the specifications and the demo program. No bugs or other problems were discovered by us.

3. **Have you implemented everything required by the programming assignment? If not, describe what are missing.**

   Yes, I have implemented everything that was required of us for #2, according to the specifications on the class website.

Section IV Self-evaluation: Points you think you deserve 6

- **Deduct one point if you submit the work after the due date but before it's closed.**
- **Grading scale**:

0. Nothing done **or missing the self-evaluation report or missing the integrity review** in the report
1. Source code is completed but the code fails to compile successfully
2. Source code can compile and do something required, but has serious bugs or miss a couple of key features.
3. Source code can compile and do most of the features required, but has many minor bugs or miss a key required feature.
4. Source code can compile and do all the features required, nearly fully functional, only a couple of minor bugs.
5. Source code can compile and do all the features required, fully functional, no bugs.
6. **In addition to the points received according to the rubrics above, get one more point** if
   a. **the self-evaluation report contains sufficient descriptions of test cases used (0.25 point)**, and
   b. **the self-evaluation report indicates the results of the test cases were verified by a peer reviewer (0.25 point)**, and
   c. **the source code is well indented and commented to make it visually very readable (0.5 point)**.