Self-Evaluation Report for Programming Assignments

Section I: Basic Information

1. **Programming assignment #: 1B**
2. **Name of the author: Esther Choi**
3. **Name of the peer reviewer, if any: Priscilla Son, Derek James**
4. **Due date of the assignment: February 15, 2017**
5. **Date when the assignment was finished: February 11, 2017**
6. **Number of hours spent in programming: roughly 4 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**:

Invalid Entry Tests: ReadDate()		
Test Case (month, day, year)	**Type of Test**	**Results/Comments**
0, 1, 2000 13, 1, 2000 -2, 1, 2000	Bad month entries (months outside 1-12 range)	Error message shows and re-entry is required until valid data is entered
1, -2, 2000 1, 32, 2000 2, 30, 2000 4, 31, 2000	Bad day entries (testing for proper number of days to each month)	Error message shows and re-entry is required until valid data is entered
1, 1, 0 1, 1, -6001	Non-positive year test	Error message shows and re-entry is required until valid data is entered
0, 0, 0 -1, 36, -1987	All bad data test	Error message shows and re-entry is required until valid data is entered
AdvanceDays()		
Test Case (month, day, year, numDays)	**Type of Test**	**Results/Comments**
2, 19, 2000, 36	Testing to see how it advances from a leap month	3/26/2000
2, 19, 2000, -36	Testing to see how it regresses from a leap month	1/14/2000
12, 15, 2000, 40	Testing to see how it advances from December into January when December is in a leap year	1/24/2001
12, 15, 2001, 40	Testing to see how it advances from December into January when December is not in a leap year	1/24/2002
1, 18, 1996, -40	Testing to see how it advances from January back into December when January is in a leap year	12/9/1995
1, 18, 1999, -40	Testing to see how it advances from January back into December when January is not in a leap year	12/9/1998
1, 18, 1997, 30	Testing from a 31-day month into a 30-day month	2/17/1997
4, 30, 1997, -50	Testing from a 30-day month back into a 31-day month	3/11/1997
7, 1, 1998, 45	Testing July to August	8/15/1998
8, 15, 1998, -45	Testing August to July	7/1/1998
1, 1, 2000, 365	Advancing 365 days from a leap year	12/31/2000
3, 20, 2000, -365	Regressing -365 days from a leap year into a non-leap year	3/21/1999
4, 20, 1999, 365	Advancing 365 days from a non-leap year into a leap year	4/19/2000
5, 31, 1999, -365	Regressing -365 days from a non-leap year	5/31/1998
6, 30, 1999, 456432	Advancing in strides (very large	2/28/3249

	number)	
6, 30, 1999, -456432	Regressing in strides (very small number)	10/29/749
BackDays()		
Test Case (month, day, year, numDays)	**Type of Test**	**Results/Comments**
2, 19, 2000, 36	Testing to see how it regresses from a leap month	1/14/2000
2, 19, 2000, -36	Testing to see how it advances from a leap month	3/26/2000
1, 15, 2000, 40	Testing to see how it regresses from January (leap year) into December (non-leap year)	12/6/1999
1, 15, 2001, 40	Testing to see how it regresses from January (non-leap year) into December (leap year)	12/6/2000
12, 18, 1996, -40	Testing to see how it advances from December (leap year) to January (non-leap year)	1/27/1997
12, 18, 1999, -40	Testing to see how it advances from December (non-leap year) to January (leap year)	1/27/2000
5, 18, 1997, 30	Testing from a 31-day month back into a 30-day month	4/18/1997
4, 30, 1997, -30	Testing from a 30-day month into a 31-day month	5/30/1997
7, 1, 1999, -45	Testing July to August	8/15/1999
8, 15, 1999, 45	Testing August to July	7/1/1999
1, 1, 2000, 365	Regressing 365 days from a leap year into a non-leap year	1/1/1999
3, 20, 2000, -365	Advancing -365 days from a leap year into a non-leap year	3/20/2001
4, 20, 1999, 365	Regressing 365 days from a non-leap year	4/20/1998
5, 31, 1999, -365	Advancing -365 days from a non-leap year into a leap year	5/30/2000
7, 30, 1999, 456432	Regressing in strides (very large number)	11/28/749
7, 30, 1999, -456432	Advancing in strides (very small number)	3/30/3249

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

The reason why I listed two peer reviewers is because I finished the program over the weekend and had my friend act as my peer reviewer (she has programming experience but it isn't her major). After Monday's class, when I saw that I had time, I asked someone from the class (someone who will likely know the program a little better) to act as my peer reviewer. Ergo, two peer reviewers.

Testing was particularly lengthy for #1B because we were running it against the demo program for verification (and we sometimes used this online calculator for verification

just to be doubly certain). We discovered no bugs or other problems during testing.

For bugs/problems I ran into while implementing the program, though, refer to my comments in DateType.cpp: IsValidDate(), lines 225-6 || IncrementDay(), lines 299-302, 309-10 || DecrementDay(), lines 339-41.

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 part 1B, according to the instructions 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)**.