Integrity rules, grading policies, and more you should know about programming assignments

1. 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 helpfully 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**: 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.

2. **Official C++ compiler version**: Visual C++ in Microsoft Visual Studio Community Edition 2019 on Windows platforms. You need to make sure your submitted programs are functional under such settings to get the points of programming assignments. **Note that a program that only works under a compiler other than our official Visual C++ version will not get the points.**

3. **Self-evaluation report, test cases, and peer review**: Note that for each programming assignment you need to fill out this self-evaluation report. In the report, you need to describe the test cases you used to verify the behavior of your program and a peer reviewer for peer review.

4. **Submission of your programming work**: For each programming assignment, (i) compress your entire Visual Studio project into a zip file and upload the zip file under Canvas and (ii) fill out the self-evaluation report and upload it under canvas

5. **Late policy**
   - **No submission accepted after the submission site on Canvas is closed**: All submissions should be done through the Biola Canvas system. No submission will be accepted after the submission site on Canvas is closed, except for extremely exceptional situations such as a serious disabling health problem with evidence from the doctors.

   - **Penalty for late submission after the due date but before the submission site is closed**: **For a programming assignment**, the submission site on Canvas may remain open for 2 more days after the due date and 1 point will be deducted from the work for a late submission before the submission is closed.

6. **Grading scale:** We'll grade each programming assignment in a 0-6 scale based on the following guidelines (could have a fraction like 5.5 points out of 6), and a one-point daily discount rate for being late.

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)**.