Code Reviews (Peer Evaluation)

One of the goals of this course is to encourage good software development practices, especially when building a large or complex software system. For each project submission, you will be assigned two other random students in the course. You must review their code and offer constructive feedback. In industry, this process is known as a *code review* and is frequently used to improve software quality and to catch software defects early.

It is important to note that the goal of a code review is to improve the quality of the **code**, not to evaluate the quality of the **developer**. Accordingly, try to use "I-messages" (e.g., "I don't understand it" rather than "it's confusing"), and ask questions and make suggestions instead of making accusations. Keep all of your comments respectful and collegial.

The following table gives examples of both *non-constructive* and *constructive* feedback (aim for the latter!) for several questions. On the next page there are checklists with more specific questions to answer during reviews.

Question	**Example of Non-constructive Feedback**	**Example of Constructive Feedback**
Is the code generally readable and well-documented?	"Not enough documentation." "I liked the author's style."	"I didn't understand the goal of function X; I think the author should add some documentation regarding its inputs and outputs." "I liked the use of extra indentation to line up the array initializations in module X."
Did you find any software defects?	"Function X doesn't work." "The program crashed on one of my test cases."	"Function X does not produce the correct output for this input: 'continue' -- I believe the regex is missing this case." "The program crashed when I tried to run it on a program with a loop. I think this is because one of the labels is being emitted in the wrong place."
What was one way that their solution differed from yours?	"My code is faster." "My function X is shorter than this author's version. Mine is better."	"I chose to calculate the maximum value on every iteration of the outer loop while this author calculated it only once and cached the result. Their approach is more efficient, but my approach works even if the list is modified during iteration."
Do you have any other constructive comments for the author?	"This code sucks and needs to be rewritten." "This code is perfect."	"Function X has redundant if-conditions; the last two could be consolidated." "The use of recursion in function Y to avoid ugly class-level data structures is very elegant."

Checklists

As you work on the project, keep this review process in mind--your code will be reviewed by at least two people in addition to the instructor! Accordingly, here is a checklist you should use before you submit:

SUBMITTER:

- ☐ The code compiles without warnings and has been tested with unit and integration tests
- ☐ The code is clean and follows a consistent coding style
- ☐ All corner cases, workarounds, and non-obvious code have been documented
- ☐ All dead code has been removed

Here is a checklist of questions to answer as you complete your peer review:

REVIEWER:

- ☐ Does the code compile? Are there compiler warnings?
- ☐ Is the code is generally readable and well-documented?
  - ☐ If not, what are some specific improvements that could be made?
- ☐ Has repetitive code been factored out? Is there any dead code?
- ☐ Did you find any software defects?
  - ☐ If so, what test case(s) exposed it? How would you attempt to fix the defect(s)?
- ☐ Are all exceptions and corner cases handled appropriately, as far as you can tell?
- ☐ Has the provided framework and existing code been used appropriately?
- ☐ What was one way that their solution differed from yours?
  - ☐ Are there any explicit tradeoffs associated with the difference(s)?
  - ☐ Now that you have seen both approaches, which approach do you prefer?
- ☐ Do you have any other constructive feedback for the author?

Code reviews should be **at least** two or three paragraphs of 3-4 sentences each, but in some cases could be much longer. Include as much detail as possible, and be as explicit as possible in your suggestions for improvement.

Submit your code review on Canvas, although you may wish to prepare your comments using a text editor or word processor beforehand to reduce the chance of data loss due to browser issues. Submit your code reviews on the appropriate assignment, but **also make sure you send each review directly** to the (correct!) author using a Canvas inbox message (CC'ing me as proof of completion).

Your review will be compared with another person's review and my own assessment, and will be graded on the following scale:

Score	**Description**
4	Thorough and constructive criticism in all areas. ***(often requires more than 2-3 paragraphs)***
3	Constructive feedback for some areas, but not thorough enough.
2	OK, but lacking (usually because of a focus on cosmetic style issues over deeper issues).
1	Mostly non-constructive or inconsistent written feedback; very superficial.
0	Nothing.