Objectives

To participate in a peer editing of a software project by following the best practices of the code review. To learn how to prepare code for review and how to be an effective reviewer. To engage in a collaborative process of code review and to ensure the reviewed code is modified according to the found errors.

Reading Assignment

To do well on this assignment, you should first read the The Best Practices for Peer Code Review. Then, you need to become familiar with How to Do Code Reviews Like a Human. Finally, to ensure a respectful environment please read The Ten Commandments of Egoless Programming.

Peer Editing Code

There are many benefits of code review, including finding defects in development and knowledge sharing that is likely to be facilitated by the process. The reviewed code will also be well-documented.

Each software innovator needs to have a portion of their software project’s code reviewed and be a reviewer for at least one author. Once the author-reviewer pairing has been established please visit the CMPSC 480 Software Innovators Google Spreadsheet with your project information. First, ensure the link to your software project is established, then add your name in the “Software Project Editor 1” column for the selected author.

Author preparation

In your software project implementation select 100-150 lines of code for review. Prepare for the review process by looking over your code and making sure you are able to explain it and to justify it. In this informal code review process, you, the author, will drive the review by sitting at the computer with a control of the keyboard and the mouse, opening various files, pointing out the code segments for review and explaining why it was done this way.

Reviewer Process

During the process of the review, if you, as the reviewer, see something amiss, you can engage in a little “spot pair-programming” as you identify errors and work with the author to write the fixes. Using Markdown, create an issue in the author’s repository, which will contain review metrics and the checklist.

Specifically, as you perform the code review, please keep track of the following, and include these metrics in your issue:

- Inspection rate: the speed with which a review is performed.
• *Defect rate*: the number of bugs found per hour of review. If you spent less than an hour on the review adjust the number to per hour rate.

• *Defect density*: the average number of bugs found per line of code.

Then, in your issue include the checklist with the following:

1. Code Formatting: following a specific style guide.

2. Coding Best Practices: no hard coding, comments, no redundancy, is the code as modular as possible, etc.


4. Code Correctness: does the code work? does it perform its intended function?

5. Code Testability: the ease of testing - refactoring into a separate function if needed.

For example, your issue may look as follows.

```markdown
## Author:
## Reviewer:
### Date of the review:
Inspection rate:
Defect rate:
Defect density:

### Checklist

- [ ] Code Formatting
- [x] Coding Best Practices
- [x] Code Readability
- [x] Code Testability
- [ ] Code Reliability
- [ ] Usability

You may include additional comments for each checkmark item or include additional items. If a specific item is not relevant to the piece of code you reviewed leave it unchecked but mention its irrelevance.

**Deliverables and Evaluation**

You are invited to submit the following materials:

1. Updated spreadsheet with relevant information.

2. Issue posted in the author’s project repository with the requirements outlined above.

3. Appropriate commit log history indicating the fix of the issues identified during the review.

Due: 15 October, 2018 at 2:30 pm