AN ABSTRACT OF THE THESIS OF

Yipeng Song for the degree of Master of Science in Computer Science presented on December 10, 2020

Title: The Analysis of Peer Reviews among First-year Computer Science College Students

Abstract approved: _____

Jennifer Parham-Mocello

In computer science, peer review, also referred to as code review, is known to be an efficient technique to ensure quality when developing software projects in various industries. Peer review is one method for encouraging computer scientists benefit from each other by providing them with the opportunity to evaluate other people's work and to receive feedback on their own work. However, this method is not commonly used among first-year Computer Science college students, due to concerns with low-quality reviews and the reliability of review scores. This study analyzes the process of peer review among first-year Computer Science colleges students by examining 1,866 completed reviews and 1,733 back evaluations to answer the following questions: 1) how accurate are first-year student peer reviews, 2) is there a correlation between review quality and student's overall performance in the course, 3) does the review score correlate the back evaluation score, 4) are differences in peer reviews related to demographics, and 5) what attitudes do students have toward peer review. The results show that high performing students provide higher quality review in design documents, and there are differences in review quality based on gender, but no significant differences based on class standing or majors. We also find that students moderately value peer reviews, and there is minimal to no correlations between review scores and back evaluation scores in Peerceptiv. These insights have implications on how peer review is executed and leaves us with open questions on how to maximize the function of peer review among first-year Computer Science college students.

The Analysis of Peer Reviews among First-year Computer Science College Students

by

Yipeng Song

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Presented December 10, 2020

Commencement, June 2021

Master of Science thesis of Yipeng Song presented on December 10, 2020

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Yipeng Song, Author

ACKNOWLEDGEMENTS

I want to express my gratitude to my major advisor, Jennifer Parham-Mocello, for providing me the opportunity of pursuing a novel line of research in Computer Science Education. Thank you for helping me troubleshoot papers, brainstorming research angles, and being a steadfast advisor.

I would like to thank the rest of my committee members, Ben Lee, Kishore Bhamidipati, and Ren Guo, each of whom has provided patience advice and guidance throughout the research process.

I would like to acknowledge everyone who participated in this research, including all consenting students, and GTAs/ULAs.

Thank you to everyone who played a role in my academic accomplishments.

Finally, a thank you to my family, who supported me with love and understanding.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

LIST OF APPENDICES

Introduction

When teaching Computer Science (CS), the main educational goals are to teach students programming concepts, to prepare students to perform professional activities, and to guide them in how to solve problems. There are different ways to reach these objectives such as having students create design, write programs, or read textbooks and research papers. Another way is to have the students read code and design by reviewing the work of their peers to practice reading code and see alternative solutions to a problem.

Peer review, also referred to as code review, is known to be an efficient technique in CS to ensure quality when developing software projects in various industries [4]. It encourages computer scientists to benefit from each other by providing them with both the opportunity to evaluate other people's work, as well as to receive feedback on their own work. However, this method is not commonly used in introductory level CS classes in college due to concerns such as low review quality and the reliability of the review scores that students give to each other. The study in this paper analyzes a strategy and multiple rubrics for incorporating and assessing peer review in a first-year, introduction to programming CS class.

There are five research questions driving this study:

RQ1: What is the review quality that first-year CS students give to each other?

It is necessary to validate the quality of peer reviews provided by first-year students for the peer review process to serve the same purpose as it does in industry. Students must provide meaningful reviews in order to have a positive impact on their peers' quality of work. If there is evidence that shows students get quality reviews through the peer review process that is similar to the grades received by the GTAs/ULAs, then this would encourage CS and many other courses outside of CS to adopt peer review into the curriculum.

RQ2: Is there a correlation between review quality and a student's overall performance in the course?

The anecdotal claim "better programmers give better reviews" is often heard through many levels of CS courses at Oregon State University (OSU), including the introductory series,

the software engineering series, and the operating systems series. Students tended to ask their instructor or teaching assistants (TAs) for review or advice on their programs or designs, as they treated them as “better programmers”. However, there is no empirical data to support this claim, and thus, it became the guiding question for this research.

At the university, the main mechanism for assessing a student’s programming ability is their grades. Since their grades reflect their overall performance in a course, the skill level of a “programmer” can be defined by their “grades”. Thus, we define a better programmer as a higher-performing student to limit the scope of this research. Having evidence that review quality does correlate to better grades could influence students to take peer review seriously and engage with it more. Finding evidence in the opposite could lead to questions of the necessity of incorporating peer review in other areas not limited to CS field.

RQ3: Is there a correlation between the peer review scores received and the rating of back evaluations?

Having the peer who receives the review judge the accuracy and helpfulness of the review, which we call back evaluation, is intended to help the reviewer provide more meaningful comments over time, and hold them accountable for the reviews they write. Since the reviews are expected to help reviewees create designs and code solutions at a higher quality level, the rating of back evaluations is based on the helpfulness and accuracy of the reviewer, without factoring in the review score given by the reviewer. Having evidence to the contrary could lead to questions on how to guide student to accept criticism, and how to increase trust between peers in the classroom setting.

RQ4: Does review quality vary based on demographics?

With increasing enrollment in the introductory CS classes, the students who take these courses are becoming more diverse. The demographics of interest in this study include gender, class standing, and majors. In this paper, we refer to both CS and Electrical and Computer Engineering (ECE) majors as “major students”, and we refer to students in majors other than ECE/CS as “non-major students”. Therefore, the following sub-research questions emerge:

RQ4.1: Do female and male students engage in peer review differently?

There is evidence to support that males and females learn differently. Females tend to rank interaction with other students higher than males, while males tend to be more externally focused, attributing their success in the classroom to external causes, such as teaching [16]. This difference may lead to female and male students to engage with the peer review activity differently.

RQ4.2: Do students with higher class standing provide better quality reviews than those with lower class standing?

Maturity can play a role in how seriously students approach their academics, which might be the same for peer reviews. Due to either a lack of maturity or not having enough prior knowledge in the subject, peer review, or an equivalent concept, is often not formally introduced until a student is in their upper course work (junior and senior level classes) at OSU. Class standing refers to the number of credits a student has earned, not their actual year in school. Some first-year students may have sophomore class standing because they took college courses or Advanced Placement (AP) classes in high school. It is rare to have third- or fourth-year CS majors take introductory courses, but when they do, it is typically to increase a grade which was formerly low or because they are switching into the major from some other major.

RQ4.3 Do ECE/CS major students give better quality reviews than non-major students?

The introductory CS courses are not limited to CS and ECE major students. Students of other majors may also take the course for different purposes. Some non-major students take this course to count in their own major, but many non-major students take it for their own interest. We are interested if the quality of the reviews by non-majors is lower than the ECE/CS majors.

RQ5: How do students' value peer review?

It is important to get students' attitude toward the process of peer review, including both reviewing others' work and getting their work reviewed by others. If students do not find peer review useful, then it cannot function as it is supposed to because students will not fully engage in the process.

Background

2.1 Course Structure

Oregon State University (OSU) is a quarter-based university. In one academic year, there are four terms or quarters: summer, fall, winter, and spring. Each quarter is 10 weeks long with an additional week for final exams.

The College of Engineering offers three main 100 level courses for Computer Science (CS) major students: CS 160, CS 161, and CS 162. CS 160 is an orientation course designed to let students explore the field of Computer Science, but it is not a prerequisite for CS 161, which introduces fundamental concepts of programming using C++. CS 162 continues to teach C++ with a focus on Object Oriented Programming (OOP), but this research study only focuses on the CS 161 course.

Topics in CS 161 involve variables and data types, conditionals, control structures, functions, one and two static and dynamic dimensional arrays, recursion, and basic memory management. In addition, students develop the problem-solving skills needed to apply the concepts they learn to programming assignments. The class has five to six assignments, depending on the instructor. The first or first two assignments focus on rote practice and the rest focus on solving real-life problems. Examples of later assignments include Battleship, Connect Four, Game of Twenty-One, and Heat Diffusion. (See Appendix B)

CS 161 is offered every term in the academic year. The prerequisites of the class are getting a grade of C or better in MTH 112, a trigonometry/pre-calculus class, or a score of 33 or higher on the Math Placement Test. Students may take CS 161 and MTH 112 at the same time. Most CS and Electrical and Computer Engineering (ECE) major students take the CS 161 course during their first year at OSU. CS 161 is a required course for both CS majors and ECE majors in the School of Electrical Engineering and Computer Science (EECS) which is why we refer to both CS and ECE majors as “major students”, and there are other students that take CS 161 because they want to learn to program or it is part of their degree requirements in majors such as Business Information System (BIS) and the Biology Genetics option which we refer to as “non-major students”.

Most major students take CS 161 during Winter term, while non-major students usually take it during the other three terms. However, some major students may not take this class because they are admitted to the university with either transfer credit from another university, or the Advanced Placement (AP) credit from high school. They also have an option to take an accelerated version online instead, but this is not advertised or recommended. Students need to get a grade of C or better in CS 161 in order to take CS 162, but non-majors might be able to count CS 161 toward their degree as long as they receive a D- or above.

CS 161 normally has around 450 students in the Winter term and around 150 students in the Fall and Spring terms. Each of the CS 161 courses is administrated by an instructor with one or two Graduate Teaching Assistants (GTAs) as well as 5-15 Undergraduate Learning Assistants (ULAs). A GTA's job is managing the ULAs, as well as helping the instructor when needed. The duties of a GTA include, but are not limited to, holding office hours, grading, leading the weekly ULA meeting and taking notes, and helping develop the labs, assignments, and exams. This is in addition to scheduling the ULAs for labs, grading the exams, and answering emails. ULAs are either sophomore, junior, or senior students, who have already taken the CS 16X courses with grades of B+ or above. The responsibilities of a ULA include leading labs, holding office hours, holding demo hours (one-on-one grading session with students, with each session lasting 10-15 minutes) for assignment grading, and answering emails. Both GTAs and ULAs are capable of helping students succeed in this course and are willing to share their knowledge. Since most of the ULAs are only one year or two further in their studies than the students who are actively taking the course, students tend to treat the ULAs as near peers.

2.2 Peer review in Assignments

In Winter 2019, there were 6 total assignments to be completed in the CS 161 course. For each assignment, students were provided with a problem statement in the assignment document. Assignment 1 and 2 were one-week assignments and only required a coding solution. Assignment 3 to 6 were two-weeks assignments and required a design document and a coding solution. In the past, students were required to submit their design documents to Canvas for peer review, but in the Winter 2019 quarter, students submitted their design and coding solutions to an online platform called Peerceptiv for peer reviews. The design document was due one week

prior to the programming assignment deadline, and the document was meant to record a student's thoughts and strategies toward solving the problem in the assignment.

The design consists of three parts based on George Polya's problem solving steps from *How to Solve It* in mathematics [12]. These steps include 1) Understanding the Problem, 2) Devising a Plan, and 3) Reflecting Back (referred to as Testing). In the first step, students are asked to rephrase the problem statement, clarify the requirements, and list assumptions if they are making any, to show they fully understand the problem. When thinking of the solution plan, students are asked to be creative. This means that they are permitted to choose pseudocode (a simplified, high-level description of the proposed algorithm or program structure), flowcharts, or anything that they think is the best method to express their thoughts, as long as they do not write actual C++ code for their plan. For the testing section, students need to submit a testing table that contains at least one good case, bad case, and edge case.

Since students only submit designs for Assignment 3 to 6, then students only peer review four designs, but they peer review five coding solutions. After submitting their work, each student reviewed two proposed designs and two coding solutions from their peers. The peers are randomly selected by Peerceptiv and the names are anonymous to the student who does the review and who receives the review. The review includes evaluating the design document, assessing the code quality, and providing constructive feedback.

The review is then judged by the peer who received the review on how accurate and helpful the review is. In Peerceptiv, this is called back evaluation, which helps the reviewer provide more meaningful comments over time by holding them accountable for the reviews they write. On the other hand, the reviews are expected to help reviewees create designs and code solutions at a higher quality level. The typical flow of tasks of one assignment is shown as below:

Design → Peer Review on design → Implement the design (coding) → Peer Review on code → Back Evaluation & Grading by GTAs and ULAs

2.3 Motivation

Peer review is known to be an efficient technique to assure quality when developing software projects in various industries [2]. In this process, one or several people will check a

program by reading parts of the source code. This collaboration of work helps to improve code quality, increase the sense of mutual responsibility, and find code defects [1].

In Winter 2019, there were more than 450 students taking CS 161 at OSU. This number is almost doubled compared to the enrollment of the same course in Winter 2015. Before Winter 2019, students submitted designs and engaged in peer review on Canvas, but the growing class size put a lot of grading pressure on the instructor and ULAs to make sure students were held accountable for quality peer reviews. In addition, coding assignments were not being peer reviewed or evaluated for quality.

Peerceptiv is a peer reviewing tool that has an algorithm for back evaluating the reviews that students provide to other students that hold students accountable for the reviews they provide, and the tool assigns a final grade to the work peer reviewed based on the back evaluations of the grading, and this grade is factored into their overall peer review grade.

When students complete their work, they are asked to review their classmates' assignment and to provide meaningful feedback. After providing feedback, reviewees are asked to judge the helpfulness of the reviews, and Peerceptiv uses this helpfulness rating, along with an accuracy score generated by Peerceptiv, to assign a back-evaluation score to each reviewer. Both reviewers and reviewees should benefit from the peer review process, especially when reviewers are held responsibility for the quality of their peer review. However, considering the course concepts are relatively new to the students, we are unsure if the peer review process serves the same purpose as it does in the industries. To better understand the function of peer reviews among college students in a first-year CS course, it is necessary to explore the peer review process in the 100-level computer science classes and to validate the accuracy of peer reviews provided by the students.

Related Work

There is a large amount of related work about the benefits of peer review in the context of computer science. However, related work on the use of peer review in the classroom setting is limited, and there are even fewer studies discussing how it functions among first-year college students. This is a complicated process and one that is not fully understood in computer science education. Not only do we need to be able to adopt a framework for executing the peer review, we also need to be able to evaluate the framework and whether or not it helps the reviewers and reviewees throughout the process.

Score or rating reliability in peer reviews is always one of the biggest concerns [6, 8, 10, 14, 15]. There is one study focused on peer assessment among higher education students [5]. It questions whether students can produce reviews comparable to the instructors and whether the misinformation will harm the students' grades. After conducting forty-eight quantitative peer assessment studies, which compared peer and teacher marks, the literature indicates that, while reliability is still an issue to consider, proper planning can overcome many of the problems. It claims that "peer assessments were found to be closer to teacher assessments when well understood criteria are used rather than assessing several individual dimensions". Although the research presented in this paper has a wider breadth than other studies, the studies presented in the current literature are very controlled and may not represent what students do in the classrooms. The studies included in this meta-analysis are not randomly allocated at the classroom level, except for one case. Thus, the result might be different if they are conducted in a classroom where the majority of students have limited programming experience. Furthermore, the current results are averaged across a wide range of outcome measures, including science project grading, essay writing ratings, and end-of-term exam scores. Aggregating across such outcomes of various topics is problematic as some measures are likely to be more sensitive to interventions than others, and therefore cannot be directly used in the computer science field.

Another study states that rubrics are important in the process of peer review [13]. This is especially true in cases where the students are not knowledgeable enough to review other's work, which is true in the study presented by this paper. Sitthiworachart and Joy noted that rubrics must be of a sufficiently large scale, i.e. 5 point vs. 3 point in the same criteria, so that the students can mark effectively during the review process. While many studies find out that the

rubrics play an important role and successfully improve the peer review activity, there is not as much agreement on how the rubrics should be implemented. The one created by Joy and Luck provides a way to select a numerical value, while others require additional comments [7]. In 1999, Mason and Woit developed a rubric that gives a list of predefined deductions and reusable comments that the students can choose from [11].

There are many studies that discuss the benefits of peer review. One research study finds that students improve their self-evaluation skills by self-evaluating through peer reviews, as they were able to determine what level of effort was acceptable from other people's work [3]. In 2004, Wolfe finds that students are able to learn from peers during the review process and get more feedback than their instructor could provide by themselves [17]. According to Wolfe, peer review not only allows the students to develop their critical reviewing skills by experiencing the other side of the grading process, but peer review allows the instructor to be able to act as a guide rather than the only source of knowledge. In the long-term setting, this should make it easier for the students to accept criticism, lead to the development of a community, and increase trust between peers in the classroom setting. Additionally, Wolfe finds a strong correlation between participation in creating peer review and the overall performance in the class. However, Wolfe did this study for a particular upper level software engineering class among only 34 students. Therefore, these conclusions are not necessarily applicable to a larger introductory Computer Science class of more than 400 students, most of whom are first-year students.

Research Method

4.1 Participants

The participants in this study are students enrolled in the Winter 2019 CS 161 class who gave consent to examine their grades and the peer reviews they gave to and received from peers. The research team taught the class, allowing for control over the implementation of assignments and rubrics for the peer reviews. Table 1 shows the demographic information of the participants, including consent rate, gender, major, and year in college.

Table 1: Demographic Details of Students

Consent Rate	166/433 (38%)
Male	136 (82%)
Female	30 (18%)
Major (CS/ECE)	121 (73%)
Nonmajor	45 (27%)
Freshman	65 (39%)
Sophomore	51 (31%)
Junior	32 (19%)
Senior	18 (11%)
Other	0 (0%)

4.2 Rubrics

For each review in this study, the participants were asked to review a randomly selected small number of their peers (at least 2) using a rubric. The peers were randomly selected by an online peer assessment tool called Peerceptiv, and the reviews and back-evaluations were anonymous. The students filled out two different rubrics the instructor created for peer reviewing design documents, and coding solutions. There was an additional rubric for the back evaluation, which was used to evaluate the quality of the review received. All rubrics assessed quality, rather than correctness, and students performed all review activities using Peerceptiv.

The rubrics for evaluating the design documents were the same for different assignments because the requirements for the design document were mostly the same (see Table 2), and they

were based on Polya's problem solving steps (full assignments for design documents can be found in Appendix B). The rating of each criteria was from 7 (Excellent) to 1 (Poor).

Table 2: Rubrics for Design Document Quality

Problem-Solving Steps 1,2,3 Did the student include 1) understanding the problem, 2) a plan/design, and 3) a testing plan?	7-The student included all 3 problem-solving steps in great detail. 6-The student included all three problem-solving steps, but they are not very thorough. 5-The student included both the understanding the problem and plan/design, but there is no testing plan. 4-The student included both the understanding the problem and testing plan, but there is no design/plan. 3-The student included a design/plan, step 2, but nothing else. 2-The student included understanding the problem, step 1, and nothing else. 1-The student did NOT include any of the problem-solving steps.
Understanding the Problem The student should address user/input requirements, assumptions, and the tasks/sub-tasks for a good quality understanding of the problem	7-The student addresses all THREE: user/input requirements, assumptions, and the tasks/sub-tasks. 5-The student addresses only TWO of the three: user/input requirements, assumptions, and the tasks/sub-tasks. 3-The student addresses only ONE of the three: user/input requirements, assumptions, and the tasks/sub-tasks. 1-The student does NOT include an understanding of the problem in their design.
Plan Quality The student should express a complete algorithm with pseudocode or a picture/flowchart.	7-The student provides pseudocode or flowchart that thoroughly explains the details of the program and both modes in the calculator. 5-The student provides pseudocode or flowchart for the choice of programmer or scientific mode, and there is a detailed outline of the steps required in one of the modes.

	3-The student provides pseudocode or flowchart for the choice of programmer or scientific mode, but there is not detail about choices and steps to perform once in these modes. 1-The student doesn't include a plan in their problem-solving steps.
Plan/Algorithm Does the plan solve the problem without any logic errors and enough detail?	7-The plan solves the problem perfectly, and there are no logic errors and plenty of detail. 5-Most of the plan solves the problem, but there are a few minor logic errors or lack of detail. 3-Some of the plan solves the problem, but there are a major logic issues or lack of detail. 1-The plan is not a reasonable solution.
Program Testing Did the paper provide a test plan with good, bad, and edge cases?	7-The design included test cases in ALL categories: good, bad, edge cases, and there were many examples in each. 6-The design included test cases in ALL categories: good, bad, edge cases, but there were NOT many examples in each. 5-The design included test cases in TWO categories: good, bad, edge cases, and there were many examples in each. 4-The design included test cases in TWO categories: good, bad, edge cases, but there were NOT many examples in each. 3-The design only included test cases in ONE category: good, bad, edge cases, and there were many examples. 2-The design only included test cases in ONE category: good, bad, edge cases, and there were NOT many examples. 1-The design DID NOT include a testing table.

The rubrics for evaluating the coding solution were separated into two sections: following instructions and code readability. The “following instructions” section was based on the core concepts that the assignment covered, and since every assignment focused on different CS concepts, the rubrics for this part were different across assignments. The “code readability” part of the rubric was mostly the same and included the evaluation of indentation/spacing, comments, and program/function headers explaining the purpose of each function. Table 3 shows an

example rubric for evaluating the code structure. Full rubrics for each assignment can be found in Appendix C. The rating of each criteria for these rubrics is from 7 (Excellent) to 1 (Poor).

Table 3: Rubrics for Code Structure of Assignment 2

Following Instructions	**Line separating each navigation** Is there a line separating each navigation and the choices are based on numbers?	7-There is a line separating each path navigation and all the choices are based on numbers. 5-Most navigations have lines separating them and most of the choices are based on numbers. 3-There are a few blank lines between navigations and some of the choices are based on numbers. 1-There are no lines separating the path navigation and none of the choices are based on numbers.
	2 paths with a depth of 3, and decisions with 2 choices Are there at least 2 paths that have a depth of 3 (an if with an if that has an if), and do all decisions have 2 choices?	7-There are two paths with a depth of 3, and most decisions have two choices. 5-There is one path with a depth of 3, and most decisions have two choices. 3-There is one path with a depth of 3, and some decisions only have one choice. 1-There are no paths with a depth of 3, and all decisions only have one choice.
	Handle bad input Does the user make sure there is an else on all decisions that handle bad number input for a choice?	7-All invalid number choices are handled as bad input. 5-Most invalid number choices are handled as bad input. 3-A few invalid number choices are handled as bad input. 1-There is no error handling for bad input.
	Create a text adventure game with element of chance	7-The adventure is implemented with if/else or switch, and there is an element of chance, and it was implemented with rand.

	Is there an element of chance in the text adventure game that is implemented with rand?	5-The adventure is implemented with if/else or switch, and there is an element of chance, but it's not implemented with rand. 3-The adventure is implemented with if/else or switch, but there is not an element of chance implemented with rand. 1-There are no if/else or switch for decisions, and there is no rand in the program for generating an element of chance.
Code Readability	**Indentation and Spacing** There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.	7-There is a consistent use of horizontal indentation with 2 spaces, 3 spaces, tabs, etc., and there is vertical spacing between themes. 5-There is some consistent use of horizontal indentation with 2 spaces, 3 spaces, tabs, etc., and there is some vertical spacing between themes. 3-There is either consistent use of horizontal indentation with 2 spaces, 3 spaces, tabs, etc., OR there is vertical spacing between themes, but not both. 1-There is no use of horizontal indentation, i.e. everything is aligned on the left, and there is no vertical spacing.
	Comments The program contains enough comments to make the program understandable	7-The program contains enough line and block comments to make the program understandable without knowing the code. 5-The program contains some line and block comments, but there are still blocks of code that are lacking explanation.

	without knowing the code.	3-The program contains a few/very little line and block comments, and much of the code is lacking explanation. 1-The program does not contain any comments.
	Program Header There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.	7-There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem. 5-There is a program header with most of the information, but there is some detail lacking in the description and input/output values. 3-There is a program header with some of the information, but there is very little detail in the description and input/output values. 1-There is no program header.

Table 4 shows the rubrics for the back evaluation. Shortly after receiving reviews from their peers, students were asked to provide feedback on the quality of each review they received. This rubric was the same for all design and code reviews.

Table 4: Rubrics for Back Evaluation

Feedback Prompt	If you found the review helpful, what made it useful? If possible, suggest a way to be more helpful next time.
Feedback Rating	5-Much More Helpful Than Average 4-Slightly More Helpful Than Average 3-Average Helpfulness 2-A little less helpful than average 1-Very unhelpful

4.3 Data Collection

We collected the Peerceptiv review ratings given and received among the consenting participants and course grade information for analysis. Table 5 shows the grade distribution of the consenting students in this class. Since grade data is categorical and peer review only accounts for 10% of the overall course grade, we use statistical tests that are nonparametric and do not assume a normal distribution. Approximately 41% of the consenting participants are "A" students, and 31% are "B" students. From the six assignments, we collected 1866 completed reviews and 1733 back evaluations. Table 6 shows the number of reviews and back evaluations collected from each peer review activity. Note that some reviews are not counted as valid reviews because the reviewed work was not graded by a GTA/ULA, due to being late or not having code that compiles.

Table 5: Distribution of Overall Course Grade of Participating Students

Course Grade	A	B	C	D	F
Percentage	41% (68/166)	31% (51/166)	14% (24/166)	5% (8/166)	9% (15/166)

Table 6: Distribution of Peer Reviews and Back Evaluations

	Asm1 Code	**Asm2 Code**	**Asm3 Code**	**Asm4 Code**	**Asm5 Code**	**Asm3 Design**	**Asm4 Design**	**Asm5 Design**	**Asm6 Design**	**Total**
Number of reviews	214	210	217	211	219	214	202	200	179	1866
Number of valid reviews	200	199	199	208	214	175	176	170	154	1695
Number of back evaluations	204	199	201	204	210	195	175	180	165	1733

4.4 Data Analysis

In this research paper, the quality of a review is measured by the difference in the assignment score given by a student compared to the score given by the GTAs/ULAs. As GTAs/ULAs are believed to be knowledgeable enough to correctly grade the assignment based on the rubrics provided by the instructor, we consider a small difference between the peer review and GTA/ULA grade to be a higher quality and more accurate review than review scores that differ greatly from the GTA/ULA grade. Furthermore, in order to avoid the cancellation of a positive and a negative difference when evaluating hundreds of data, the absolute value of the difference is used, and this value is called delta.

We collected three sets of data to evaluate RQ1 and RQ2: 1) the peer review scores given by the consenting participants, 2) the actual assignment scores given by the GTAs/ULAs, and 3) the overall course grades of the consenting students. We evaluated RQ1 by calculating the average delta value of the class. We evaluated RQ2 by first dividing the students into different groups bases on their overall course grades and then comparing the delta values of these groups. We evaluated RQ3 by correlating the rating of back evaluations and the peer review scores received. We stratify the data based on demographic information from the Registrar and compare the delta values within different demographic groups to answer RQ4, and we use the two questions in each assignment survey to answer RQ5. An analysis from each evaluation of the five research questions is presented in the following chapter.

Results and Discussions

RQ1: What is the review quality that first-year CS students give to each other?

Since students in this research study review design documents and coding solutions, we analyze the review quality of coding solutions and the review quality of design documents to answer this research question.

Table 7 shows that the average delta for the coding solutions is 12.24, with a negligible correlation to the actual grades given by the GTAs/ULAs (with an average correlation coefficient value of 0.27). The result indicates that, for a 100-point coding assignment, the average difference in scores given by the first-year CS students and the GTAs/ULAs is 12.24. However, it is interesting to note that the average delta and the correlation coefficient increases over time, which suggests that the review score and actual grade received become inversely correlated over time.

The small correlation coefficient or increase in correlation with a larger delta might suggest that the code reviews first-year CS students give to each other may not be reliable. However, we need to further analyze the relationship between the review quality of coding solutions and students' overall performance in RQ2 to have a better understanding on whether this result applies to all students in the class or if there is a difference in review quality between higher-performing and lower-performing students.

Table 7: Coding Solution Review Scores Detail

	#of valid reviews (total 1020)	Avg. review score (100%)	Avg. Asm score (100%)	Avg. delta	Correlation coefficient value
Code 1	200	90.74	94.66	8.79	0.17
Code 2	199	91.49	98.53	11.07	0.27
Code 3	199	91.39	84.13	12.68	0.21
Code 4	208	90.52	82.39	13.83	0.39
Code 5	214	92.28	89.72	14.85	0.31
Average	204	91.28	89.89	12.24	0.27

According to Table 8, the average delta for the design documents is 6.20, meaning that for a 100-point design document, the average difference in scores given by the first-year CS students and the GTAs/ULAs is 6.20. It also suggests that the review scores and the actual grades are moderately correlated (with an average correlation coefficient of 0.49). Additionally, with the decrease in average delta in later designs, the review scores given by peers became more reliable and directly correlate to the grades given by GTAs/ULAs. This may be because students became more knowledgeable as the term progresses, or because they are benefitting from the process of peer review. When students review another person's design, they are evaluating the quality of the work, but they are also analyzing the choices and structures made by the peers. In addition, they see some of the concepts they have been taught in class and alternative solutions expressed in different ways. Not only do students learn from the assessment of the work, but they learn from seeing new ideas, choices, and techniques (good or bad) in the review process.

Table 8: Design Review Scores Detail

	#of valid reviews (total 675)	Avg. review score (100%)	Avg. Asm score (100%)	Avg. delta	Correlation coefficient value
Design 3	175	92.36	97.33	7.29	0.30
Design 4	176	92.27	98.28	7.12	0.49
Design 5	170	94.12	98.54	5.46	0.56
Design 6	154	94.21	96.31	4.94	0.63
Average	168.75	93.24	97.62	6.20	0.49

RQ2: Is there a correlation between review quality and a student's overall performance in the course?

As the reason explained in the RQ1, we divide this research question into two sub-questions: 1) do better students provide better review quality for coding solutions, and 2) do better students provide better review quality for design documents.

Figure 1 shows the average delta values between the five coding solutions and students' final grade in the class. From the diagram, we do not see that higher performing students (students with A or B grades) have lower delta values than lower performing students (students with C, D, or F grades). This may indicate that the review quality for coding solutions does not correlate with the overall performance of a student in the course, which might be due to how the coding solutions are graded by GTAs/ULAs. In peer review, students score the coding solution using a rubric focusing on the quality of the program, such as spacing/indentation, function/program headers, and comments, while the GTAs/ULAs grade the same coding solution using a rubric focusing on the correctness and functionality of the program with only a small portion focusing on quality. Therefore, it is not surprising to see that the two scores are not related for the coding portion of an assignment.

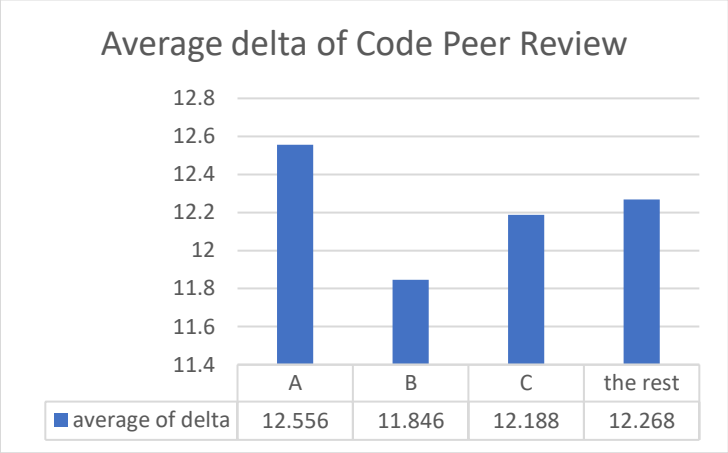

Figure 1: Average Delta of Code Peer Review vs. Student Performance

Furthermore, these results might suggest that a fully functional coding solution may have a low-quality code structure. As educators in CS, the general goal is to provide students with a systematic approach to solve problems. A systematic approach to developing software should lead to a both functional and readable program. In the initial stages of CS, the focus of solving the problem is often on creating a correct solution using a particular programming language, while the emphasis on the quality and efficiency of the solution is largely missed. Considering the low correlation in the two sets of scores shown in the CS 161 class of Winter 2019, we need to develop a different rubric that grades on both correctness and the quality of the coding solution to gain a more accurate view of how well a student's program is written, as well as hold our students accountable for well-written solutions.

Figure 2 shows the average delta values between the four design documents and students' final grade in the class. The diagram indicates that higher performing students (students with A or B grades) do provide higher quality reviews compared to the lower performing students (students with C or D or F grades). Since the GTAs/ULAs grade the design documents on quality rather than correctness, it makes more sense that these sets of scores would better align. This result suggests that better students provide better reviews on design documents. This is likely because higher performing students usually have a better understanding of the assignment requirements compared to lower performing students, and thus can provide constructive feedback to others.

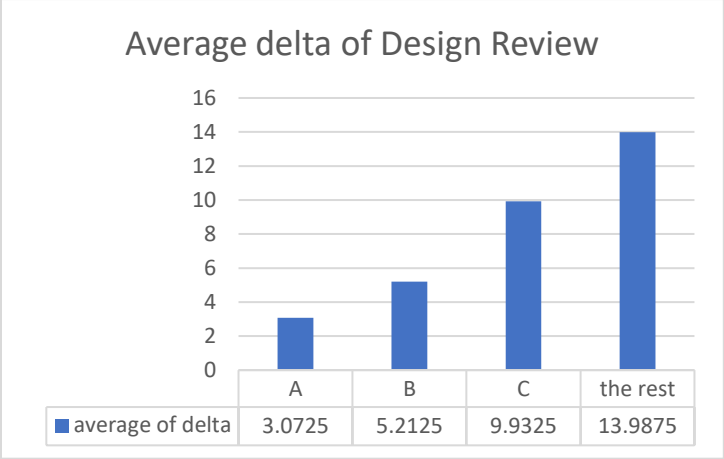

Figure 2: Average Delta of Design Review vs. Student Performance

RQ3: Is there a correlation between the peer review scores received and the rating of back evaluations?

According to Figure 3, there is minimal to zero correlation between the peer review scores received and the rating of back evaluation (average correlation coefficient less than 0.03). This indicates that students were not judging the helpfulness or the accuracy of the received peer reviews by the review scores. We further evaluate the correlation between the rating of back evaluation and the length (number of words) of the constructive feedback in reviews. The average correlation coefficient is 0.3, much higher than with the review scores. While the longer feedback is not necessarily more helpful or more accurate than the shorter one, it contains more information in most of the cases. Therefore, it is not surprising to see a higher correlation

between the rating of the back evaluation and the length of the review than the review scores. This result aligns with other research that indicates that peer review allows the students to accept the criticism and increase trust between peers in the classroom setting.

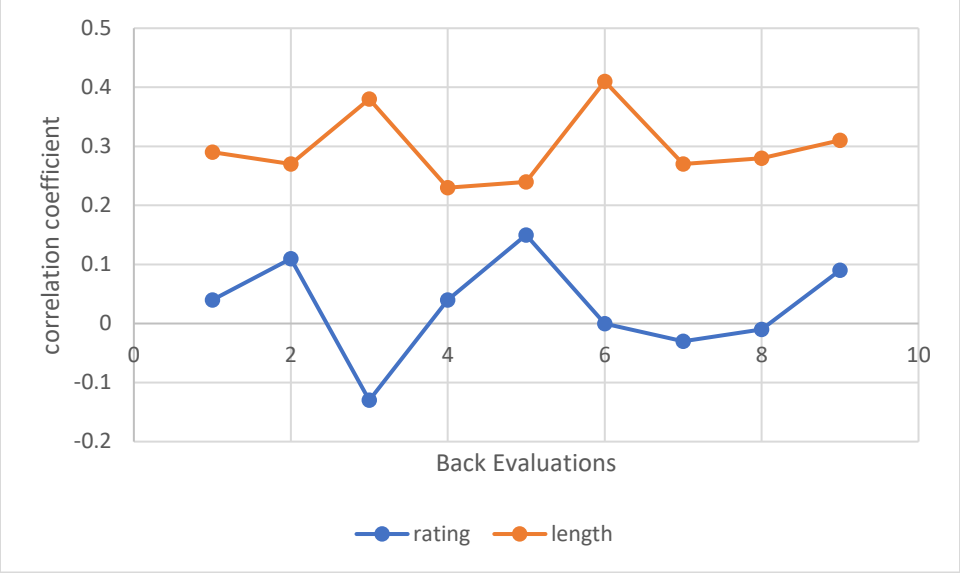

Figure 3: Coefficient Values of Back Evaluations and Review Scores

RQ4: Does review quality vary based on demographics?

Since the review quality cannot be judged based on the delta values of the coding solutions, due to reasons explained in RQ1 and RQ2, then we only analyze the delta values from the design documents to answer this research question and the sub research questions. Since the delta values are not normally distributed, we use the Kruskal-Wallis statistical test to evaluate if there is a significant difference between the two groups, which are divided by the demographic factor, using a 95% confidence interval. The null hypothesis is that there is no difference in delta values between the two groups.

RQ4.1: Do female and male students engage in peer review differently?

Table 9 shows the statistical test result of female and male design review quality. With the p-value less than 0.05 (p = 0.02), we reject the null hypothesis, and state that there is a significant difference in female and male review quality. Further analysis shown in Figure 4 and 5 suggests that female students have an average delta value of 4.38 with a correlation coefficient of 0.81. Whereas male students have an average delta value of 6.45 with a correlation coefficient of 0.45 on design reviews. This result suggests that female students give higher quality reviews

than the male students, at least in the Winter 2019 CS 161 class. This aligns with the results found by Wehrwein, Lujan, and DiCarlo in 2007, which also observed that female students performed better in interactions with other students while learning [16].

Table 9: Test Result of Female and Male Design Review Quality

Row Labels	**Sum of rank**	n1 (female)	27
female	1500.5	n2 (male)	117
male	8939.5	H	5.47
Grand Total	**10440**	chi-square	3.84
		p-value	0.02

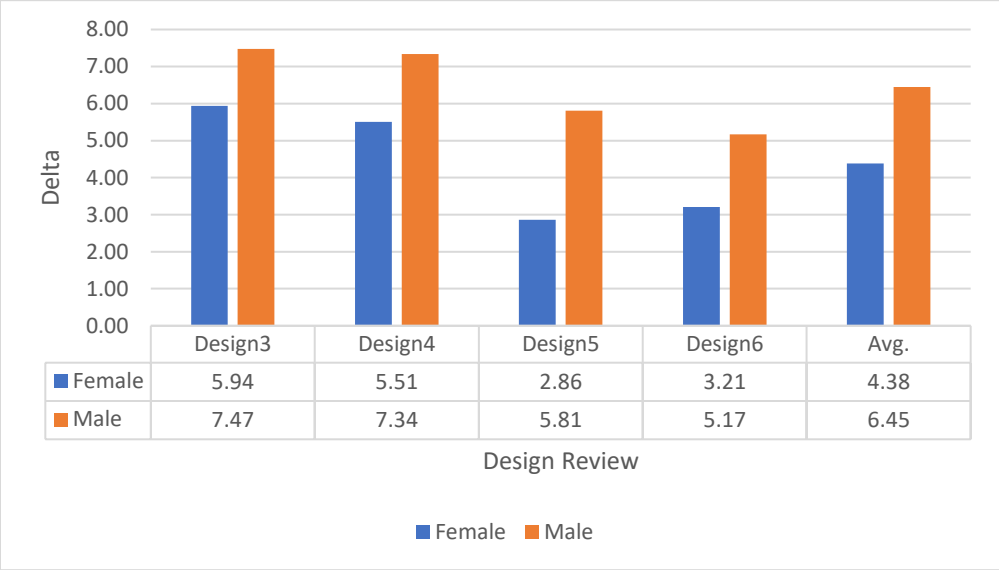

Figure 4 Design Review Delta of Female vs. Male Students

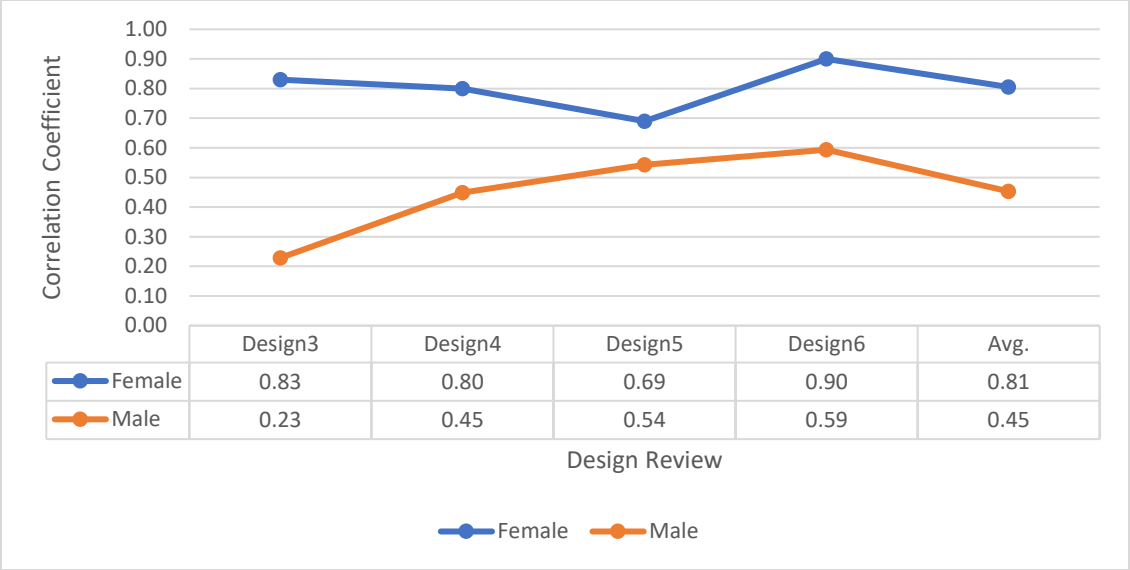

Figure 5 Design Review Correlation Coefficient of Female vs. Male Students

RQ4.2: Do students with higher class standing provide better quality reviews than those with lower class standing?

While the p-value of 0.16 indicates that there is no significant difference between the two groups (see Table 10) Figures 6 and 7 show that the correlation coefficient is greater among students with a higher class standing (0.73 vs. 0.49), and students with a higher class standing (juniors and seniors) have a smaller delta value (4.89) than the students with a lower class standing (freshman and sophomore) (6.48). Even though the p-value suggests that a certain level of maturity and experience is not necessarily needed in order to do the peer review well in the Winter 2019 CS 161 class, students with a higher class standing give significantly more quality reviews than students with a lower class standing for the last two assignments, which might indicate that maturity plays a role in review as the complexity of the assignment increases.

Table 10: Test Result of Higher- and Lower-Class Standing Design Review Quality

Row Labels	**Sum of rank**	n1 (higher)	44
higher	2864.5	n2 (lower)	100
lower	7575.5	H	1.99
Grand Total	**10440**	chi-square	3.84
		p-value	0.16

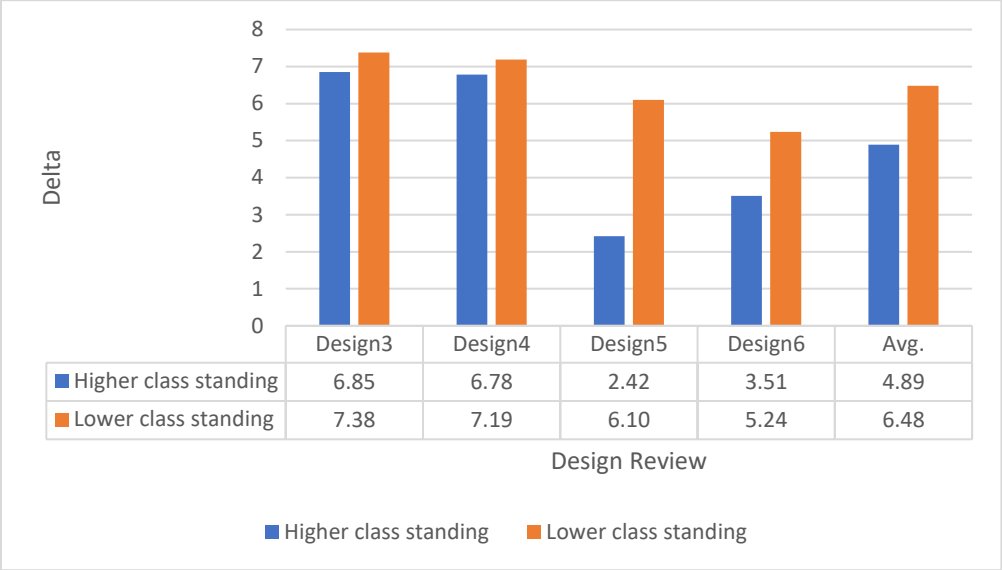

Figure 6 Design Review Delta of Higher vs. Lower Class Standing Students

Figure 7 Correlation Coefficient of Higher vs. Lower Class Standing Students

RQ4.3 Do ECE/CS major students give better quality review than non-major students?

As with class standing, the p-value for the statistical test comparing students majoring in ECE/CS with those who are not is greater than 0.05 (p=0.49), which means that there is no significant difference in the review quality between major and non-major students (see Table

11). Figures 8 and 9 also show that the average review quality of the major students is only slightly higher than that of the non-major students with average delta values of 5.26 and 6.87 and average correlation coefficient values of 0.52 and 0.38, respectively.

Table 11: Test Result of Major and Non-major Design Review Quality

Row Labels	**Sum of rank**	n1 (major)	105
major	7458.5	n2 (non-major)	39
non-major	2981.5	H	0.48
Grand Total	**10440**	chi-square	3.84
		p-value	0.49

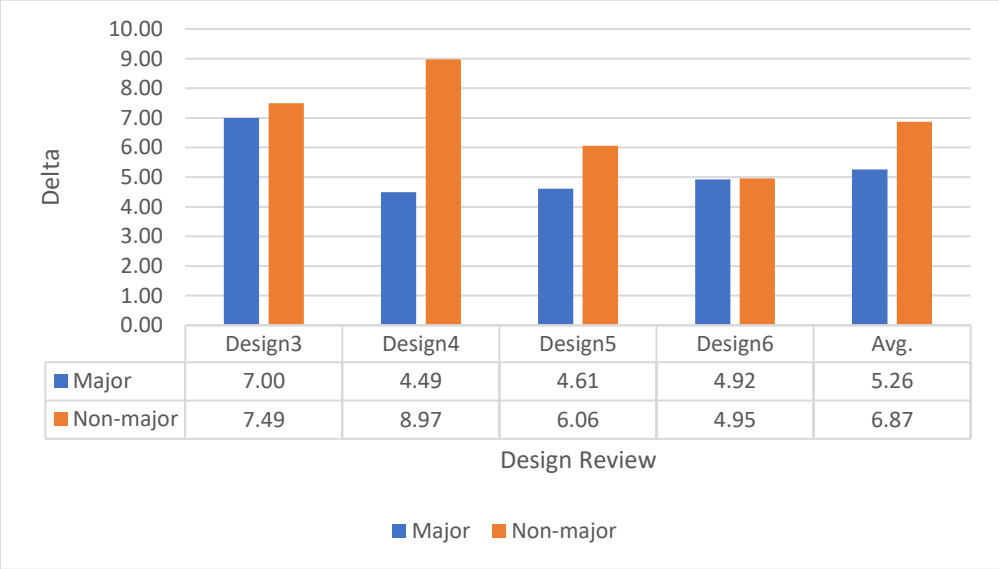

Figure 8 Design Review Delta of Major vs. Non-major Students

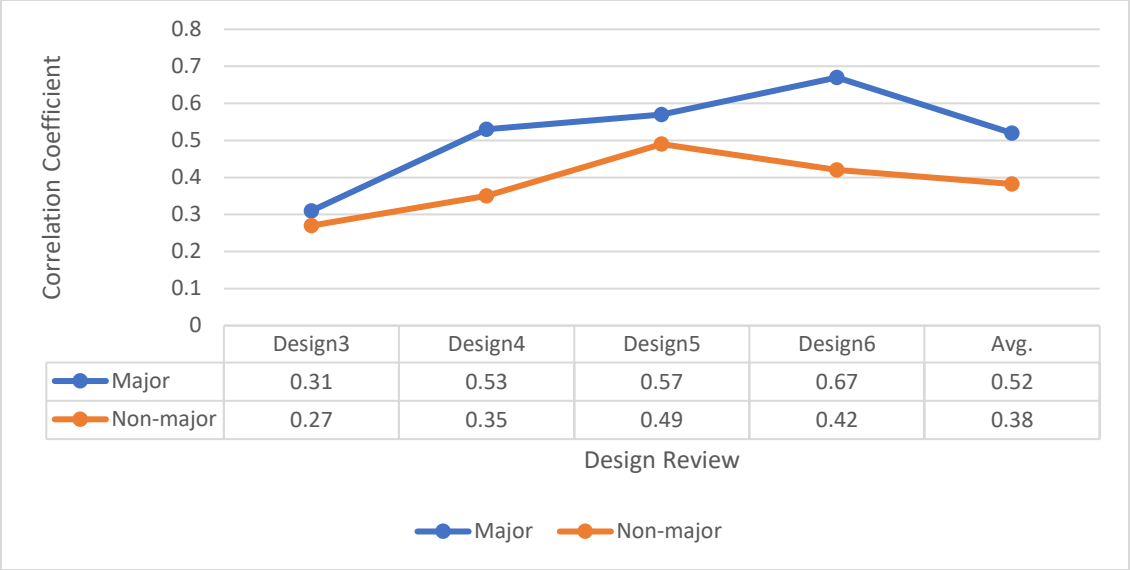

Figure 9 Correlation Coefficient of Major vs. Non-major Students

RQ5: How do students' value peer review?

For assignments 1-5, we asked students to fill out a survey about their experience with the peer review activity. We use responses to two of the survey questions to evaluate RQ5, which are listed below:

Q1: Overall, how useful did you find reviewing your peers' assignments to be?

1-Extremely useful 2-Very useful 3- Moderately useful 4-Slightly useful 5-Not useful

Q2: Overall, how useful did you find receiving reviews to be??

1-Extremely useful 2-Very useful 3- Moderately useful 4-Slightly useful 5-Not useful

According to Table 12, Table 13, and Figure 10, the survey has a response rate of around 85% (709/830). However, only 7% (49/709) of the participants think that having their work reviewed is "extremely useful", and about the same proportion (46/709) of the participants think it is "extremely useful" for reviewing their peers' work. On average, in Winter 2019 CS 161,

consenting students rate both having their own work reviewed and reviewing others' work to be "moderately useful" (3.11 vs. 3.24), but students rate having their own work reviewed slightly more useful than reviewing others' work. In general, students agree that doing a peer review is useful. However, the improvement seems marginal, at least according to the students. This could be due to the low quality of the reviews most students receive or because the peer review activities are based on the quality of the work, while the majority of their course grade is based on correctness. In other words, the improvement of the coding solution or design document quality might not be deemed as important, since it is not a large portion of their grade. Therefore, students may not see the improvement gained from the peer review process reflected in their grade, and thus did not give a positive rating to the usefulness of peer reviews.

Another interesting observation is that peer review ratings increase in later assignments, meaning students found peer reviews less useful as the term continued. This is likely due to the increasing complexity of later assignments and the freedom allowed in programming the various solutions for those assignments, as there are many more ways to solve assignments 4 and 5 than assignments 1 through 3. It is possible that once a student finds their own solution, they might give low grades to everything else or refuse to accept a different solution written by their peers. This hurts the functionality of the entire peer review process, making it less useful than it is supposed to be. Another potential reason for these results might be due to how peer review grades are factored into the overall course grade. As the term progressed, students might have realized that the peer review grade was based on completeness rather than correctness and therefore, this affected the quality of the reviews, thus making them less useful.

Table 12: Q1 Response Detail

	asm1	asm2	asm3	asm4	asm5	Total
Total Response	128	142	148	147	144	709
Extremely useful	9	13	12	5	10	49
Very useful	33	37	30	31	22	153
Moderately useful	50	49	58	60	56	273
Slightly useful	27	21	27	30	35	140
Not useful	9	22	21	21	21	94

Table 13: Q2 Response Detail

	asm1	asm2	asm3	asm4	asm5	Total
Total Response	128	142	148	147	144	709
Extremely useful	11	12	10	4	9	46
Very useful	28	26	28	21	23	126
Moderately useful	40	57	53	57	44	251
Slightly useful	39	29	37	43	39	187
Not useful	10	18	20	22	29	99

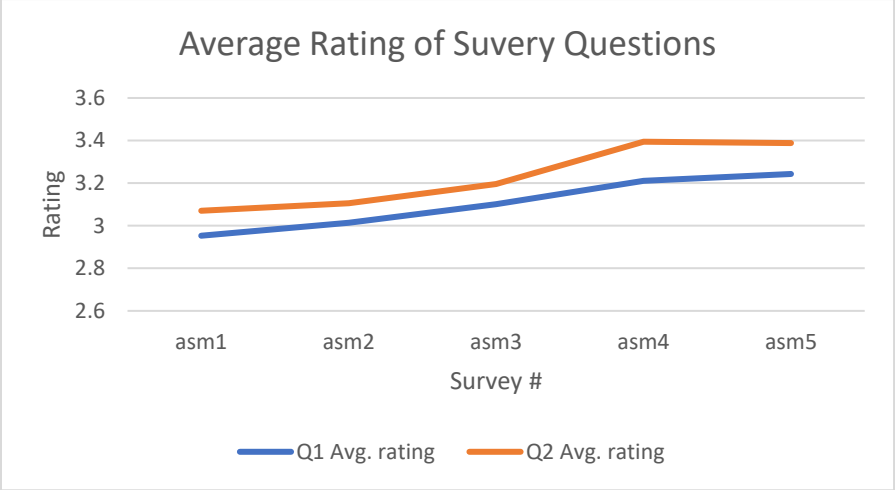

Figure 10: Average Rating of Peer Review Survey Questions

Conclusion, Threats, and Future Work

This research study presents one method of executing and evaluating the process of peer review among first-year CS college students. While the study is unable to present data from other first-year students taking different CS classes, the results are still valuable. We find that the peer code review is not correlated to the grade received on the coding solution. Whereas, the peer design review is much more correlated to the actual grade received on the design, and this correlation becomes stronger as the term progresses, which means that the students are becoming more reliable and giving more similar review scores as the GTA/ULAs over time. We also see that students with better performance in the Winter 2019 CS 161 class provide high quality reviews for design documents. While the study is unable to judge the function of reviewing coding solutions among these students because of the difference in rubrics used in the peer review and actual grading, i.e., one on quality and one on correctness, there are still important lessons learned from these results. The quality of the coding solution, or coding structure and efficiency does not play an important role in the grade received for a programming assignment, which means that code is primarily graded on correctness. Further work on how to balance grading programming assignments for correctness and quality will help develop a much more balanced rubric and grade distribution, as well as hold students accountable for "good" code.

This study shows minimal to zero correlation between back evaluation ratings and the review scores received, indicating that students went over the comments left by the reviewers and did not select the back evaluation rating based on the review score given. We believe this helps students with their critical thinking by reviewing the feedback and comparing it to their own understanding and/or the requirements in the assignment documents to determine if the peer review is valid and worthwhile.

We also see that some demographic factors affect the review quality. Female students provide better quality reviews that correlate more to the actual grades given by the GTAs/ULAs. We do not see a significant difference in review quality based on whether a student is major in ECE/CS or has a higher class standing (junior or senior). However, since the number of female students and students with a higher class standing are relatively small in this study, we need to verify this with additional data in a future study.

Lastly, students think the process of peer review is moderately useful, and work is ongoing to examine a refined peer review rubric to let first-year student value peer review more, as well as incorporate the assessment of quality into the rubrics for the GTAs/ULAs. We understand that all results from this research study may not be representative to other classroom settings. However, this paves the way for other researchers to investigate the use of peer review in introductory CS class, as well as Peerceptiv as a peer review tool in CS courses.

References

[1] Bacchelli, A, and C. Bird. Expectations, Outcomes, and Challenges of Modern Code Review. *ICSE,* 2013. https://sback.it/publications/icse2013.pdf

[2] Baum, T., O. Liskin, K. Niklas, and K. Schneider. Factors influencing code review processes in industry. *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* November 2016 Pages 85–96 https://doi.org/10.1145/2950290.2950323

[3] Davies, R. and T. Berrow. An Evaluation of the use of computer peer review for developing higher-level skills. *Computers in Education* 30(1/2): 111. 1998.

[4] dos Santos, E.W., Nunes, I. Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. *J Softw Eng Res Dev* **6,** 14 (2018). https://doi.org/10.1186/s40411-018-0058-0

[5] Falchikov, N., and J. Goldfinch. Student Peer Assessment in Higher Education: A Meta-Analysis Comparing Peer and Teacher Marks. *Review of Educational Research,* Sept. 1, 2000. https://doi.org/10.3102%2F00346543070003287

[6] Gehringer, E. F. Electronic peer review and peer grading in computer-science courses. *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*. 2001, Charlotte, North Carolina, United States, ACM Press. 2001.

[7] Gehringer, E. F. Electronic peer review builds resources for teaching computer architecture. *Proceedings of the American Society for Engineering Education Annual Conference & Exposition,* American Society for Engineering Education, 2003.

[8] Hamer, J., K. T. K. Ma, et al. A method of automatic grade calibration in peer assessment. *Proceedings of the 7th Australian conference on Computing education* - Volume 42. Newcastle, New South Wales, Australia, Australian Computer Society, Inc 2005.

[9] Joy, M. and M. Luck. Effective electronic marking for on-line assessment. Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education, ACM Press, 1998.

[10] Liu, E. Z. F., S. S. J. Lin, et al. Alternatives to instructor assessment: a case study of comparing self and peer assessment with instructor assessment under a networked innovative assessment procedures, *International Journal of Instructional Media*. 2002.

[11] Mason, D. V. and D. M. Woit. Providing mark-up and feedback to students with online marking. The proceedings of the thirtieth SIGCSE technical symposium on Computer science education, ACM Press, 1999.

[12] Polya, G., 1957. How to Solve It: A New Aspect of Mathematical Method, Princeton, NJ: Princeton University Press

[13] Sitthiworachart, J. and M. Joy. Web-based peer assessment in learning computer programming. *Proceedings 3rd IEEE International Conference on Advanced Technologies*, July, 2003, 9-11. https://ieeexplore.ieee.org/document/1215052

[14] Tseng, S.-C. and C.-C. Tsai. On-line peer assessment and the role of the peer feedback: A study of high school computer course. *Computers & Education* 49(4): 1161-1174, 2007.

[15] van Hattum-Janssen, N. and J. M. Lourenco. Peer and Self-Assessment for First-Year Students as a Tool to Improve Learning. *Journal of Professional Issues in Engineering Education & Practice* 134(4): 346-352, 2008.

[16] Wehrwein, E. A., Lujan, H. L., DiCarlo, S. E. Gender differences in learning style preferences among undergraduate physiology students. *Advances in Psychology Education*, 01 JUN 2007 https://doi.org/10.1152/advan.00060.2006

[17] Wolfe, W. J. Online student peer reviews. In Proceedings of the 5th conference on Information technology education, ACM Press. 2004.

Appendix A

Syllabus of Winter 2019 CS 161 Course

Office Hours - 2101 KEC

- Dr. Jennifer Parham-Mocello (2021 KEC): M 11am-12pm, 3pm-4pm & W 11am-12pm

Class Information

- Class Email - cs161-001-w19@ENGR.ORST.EDU (distributed to whole class, TAs, and instructor)
- TA Email - cs161-ta@ENGR.ORST.EDU (distributed to all TAs and instructor)
- DO NOT send messages through Canvas. Send emails directly to myself, TAs, or class using the above addresses.

Course Material

There is no book for the course, but you are required to purchase a seat in Peerceptiv for $12.50. This tool is used to enhance your learning by reviewing peer work and receiving peer reviews. You can create a new account on Peerceptiv by going to https://www.go.peerceptiv.com.
Our class code is: **happen12**

Textbook Resources

- (Wikibooks) Wikibooks C++ Programming
- (Miller) An Introduction to the Imperative Part of C++
- (Downey)How to think like a computer scientist

Additional Textbook References

Author(s): Gaddis T., Walters, & Godfrey (2014)
Name: Starting out with C++ : early object, 8th Edition
Publisher: Boston, MA: Addison-Wesley.
Digital ISBN: 978-0133449198
Paperback ISBN: 978-0133360929

Author(s): Schildt, H. (1998)
Name: C/C++ Programmer's Reference, 3rd Edition
Publisher: Berkley, CA: Osborne McGraw Hill
Digital ISBN: 0-07-213293-0
Paperback ISBN: 0-07-882476-1

Author(s): Savitch, W. (2012)
Name: Absolute C++ W/ MyProgrammingLa, 5th Edition
Publisher: Boston, MA: Addison-Wesley
Digital ISBN: 978-0132846578
Paperback ISBN: 978-0132989923

Author(s): Dale, N. & Weems, C. (2009)
Name: Programming and Problem Solving with C+, 5th Edition
Publisher: Jones & Bartlett Publishers
Paperback ISBN: 978-0763771560

Prerequisites

MTH 112* [C] or Placement Test MPT(33) or Placement Test MPAL(061)

Course Description

Overview of fundamental concepts of computer science. Introduction to problem solving, software engineering, and object-oriented programming. Includes algorithm design and program development.

Course Content:

- Identifiers and primitive data types
- Assignment, arithmetic, logical, and relational operators
- Expressions and statements
- Flow of control: selection, repetition, recursion
- Functions/parameter-passing including call-by-value and call-by-reference
- 1- and 2-dimensional arrays, strings, and other structured data types
- Pointers
- Error Handling
- Debugging

Course Objectives

1. Design and implement programs that require
    a. various control statements involving selection and repetition
    b. expressions with variables, constants, function calls, pointers, and arithmetic/relational operators with mixed data
    c. arrays, strings, and other data structures
    d. library functions and programmer-defined functions with parameter-passing by value and by reference
    e. abstraction, modularity, separation of concerns
    f. use of the object-oriented programming model
2. Debug programming syntax and run-time errors.
3. Produce recursive algorithms, and choose appropriately between iterative and recursive algorithms.

4. Describe and apply basic software engineering design principles and software quality factors.

A detailed description and time-line of the topics covered in this course can be on the calendar page. This calendar is strictly a guide for the course. It is tentative and subject to change. You can find the topics covered in the daily lectures in slides posted on the calendar page, and the assignments with their corresponding actual due dates are located on the assignments page.

Attendance Policy

- Lecture: Strongly Encouraged. In class extra credit exercises can only be made up with approved absences.
- Labs: Required. You must attend your registered lab section. Missed labs result in a zero score for the lab and cannot be changed without an excused absence. A lab may be excused by emailing your TAs and instructor at cs161-ta@engr.orst.edu BEFORE the end of lab with the following pieces of information
  1) the subject line is"Missing 161 lab"
  2) the lab you are missing
  3) your excuse
  4) your plan for making up the lab, which can be the other lab you plan to attend in the week or doing the lab on your own prior to the next lab.
  A valid excuse includes, but not limited to, family emergency, injury, hospitalization, death, birth of a child, trauma, illness, or school-related (or other approved) events. The TAs in the lab you are missing or instructor will verify the message is BEFORE the end of the lab and excuse, and we will or will not provide consent to the absence. All decisions made are final.
- If the instructor is late for a lecture or a TA is late for lab, please remain in the classroom for 10 minutes.
- You must remain in the lab until your work is complete or until the lab period has ended, as well as **viewed/graded** by a lab TA.

Technology Requirements

- Laptops required in lab. Make sure they are fully charged as outlets are limited.
- Laptops are not welcome in lecture. If you must use a laptop, you are required to sit in the back rows of the class room. If there are extenuating circumstances that require the use of a laptop in class, arrangements need to be made with the instructor prior to attending lecture.
- Cell phones, tablets and other mobile devices should be silenced and away during lecture. Repeated use of a cell phone or tablet in lecture will result in you being excused from the lecture.

About My Courses

- BE PROACTIVE, don't be reactive!!!

- BE RESPECTFUL, no one is perfect. Programming is difficult, and everyone in the class comes with different skills. The brain is a muscle and needs a work out in this area. You weren't born programming, you learned with practice. Please read the document on establishing a positive community.
- HAVE A GROWTH MINDSET!!! Everyone around you does not know more than you:) On the other hand, you do not know everything!
- "For every teacher there are two learners." This is a learning environment for all skills. You can learn from helping others.
- ACTIVE LEARNING! I need YOU to participate.
- "...it is far more honorable to fail than to cheat."
- "The task of the modern educator is not to cut down jungles but to irrigate deserts."

Grade Evaluation Scores for labs, assignments, and exams will be posted on Canvas as they are graded.

Labs - 20%

- You are **required to bring a laptop to the lab**.
- There are 10 total labs in this course, i.e. one to be completed each lecture week. Since Monday labs are cancelled in week 3, we will have a take-home lab that you can complete on your own or go to a Tuesday-Friday lab.
- You MUST attend the lab in which you are registered, unless receiving permission PRIOR to missing the lab!!! (Read Attendance policy!)
- Some parts of the lab will be group work while others will be individual work. You can submit one copy for any portion of the lab that is group work.
- Labs are graded on a 10-point scale broken down into any combination of quiz questions, hand-written responses to conceptual questions, and hands-on coding.
- These labs are supposed to enhance the lectures using reflective reasoning and hands-on learning.
- If you attended a lab and did not finish, you can finish **up to 3 points of the lab** at home and bring it with you to the next week's lab for a grade. You must show your update from the previous lab within the first 15 minutes of the current lab to redeem points on the prior lab, otherwise the work is not accepted.
- If you have a problem with a lab grade, **you must contact your TAs through EMAIL within ONE WEEK** of receiving your grade. After one week, you will not be able to dispute your grade.

Peer Reviews - 10%

- You will assess the quality of your peers' designs or programs each week on Peerceptiv. Read the instructions above under "Course Materials" to create an account.

- There are 9 total peer reviews to be completed over the course of this class. Assignment 1-5 will receive code reviews, and assignments 3-6 will receive design reviews.
- You will provide meaningful comments that will be judged by your peers on Peerceptive.
- Your peer review grade is a combination of doing your peer reviews, how accurate and helpful your reviews are, and how your peers rate the quality of your work.
- If you are not going to submit a design or program for an assignment, then you can confirm this in Peerceptiv and still participate in reviews. Do not click this if you are submitting a program late because you are not allowed to submit an assignment after you have reviewed others!!!

Assignments - 30%

- There are 6 total assignments to be completed over the course of this class. Assignments 1-2 will only include code, and assignments 3-6 will include a design and code.
- DO NOT expect answers to emails about assignments after 5pm on the day it is due.
- All assignments include designing or writing a computer program, and programming assignments MUST compile and execute on ENGR.
- Programming assignments that do not compile on ENGR will receive a grade of **zero on the implementation portion of the assignment without any exceptions.**
- Assignments are to be turned in before Midnight (by 11:59pm) on Sunday night, otherwise the assignment is late.
- You will turn in your assignments using Peerceptiv. Read the instructions above under "Course Materials" to create an account
- Programs are evaluated on how well they solve the assigned problem (adherence to program specification), proper formatting/use of comments, and creativity.
- You start with 5 points extra credit on your assignment total. **You can use these points for turning in your coding portion of the assignment(s) late, and each point is equal to one day**. You can use these extra credit points on one program or spread them out across programs. Absolutely no designs are accepted late, and failure to submit a design or program will affect your peer review grade. A program will be accepted without penalty as long as it is submitted prior to the number of extra credit points you have remaining. After using your 5 extra credit points, a late program will not be accepted, and a late design is never accepted. You turn in a late program to Peerceptiv just as you would normally.
- Assignments in this course are graded by demoing your work for 15 minutes with a TA. You are required to **meet with a TA** within two weeks of the due date to demo. You can schedule a demo with a TA on the home page in the far right column of the bottom table labeled "Grading Hours".
- **Demo Outside 2 Weeks:** Assignments that are not demo'd within the acceptable time period will be subject to a 50 point deduction.

- **Demo Late Assignments:** Late assignments must still be demoed within the two week demo period beginning from the assignment's due date.
- **Missing a Demo:** If you miss your demo with a TA, you will receive a 10 point (one letter grade) deduction to that assignment for each demo missed. If you need to reschedule a demo, remove your name from the poll before rescheduling. If you need to **reschedule the day of your demo**, still remove your name from the poll and email your demo TA with subject **"CS 161 Cancel Demo"**.
- If you have a problem with an assignment grade, **you must contact your TAs through EMAIL** within ONE WEEK of receiving your grade. After one week, you will not be able to dispute your grade.
- Remember to use your TAs because they are the ones who execute, read, and grade the assignments.

Exams - 30% (15% each exam)

- The are 2 total exams for this course.
- The exams are true/false and multiple-choice, and they will be given during lecture time.
- A student must notify the instructor BEFORE the exam to schedule a make-up.

Final Exam- 10%

- There will be a cumulative final exam.
- The final exam may include any combination of programming, written work, and explanation of existing code.
- A student must notify the instructor BEFORE the final to schedule a make-up.

Proficiency Demo (Ability to Keep a Passing Grade (C or above)!)

- You will take a pass/fail live proficiency coding demo during the lab in week 10.
- There will be a practice demo given in week 5 with material from the first half of the class. In the practice demo, you will be scored on a pass/almost/no-pass score to give you feedback on where you stand in the class.
- If you do not have a passing grade in the class (below C), then failing the proficiency demo cannot hurt your grade because you are not expected to be proficienct (C or above) in the class to move forward. However, if you have a passing grade in the class (C or above),then you are expected to pass the proficiency demo to keep your passing grade in the class, i.e. **you cannot receive higher than a C- without passing the proficiency demo.**
- Do not freak out!!! If you have a passing grade for your assignments and YOU have been the one to do the work, then you should be able to pass the proficiency demo program.
- A student must notify the instructor BEFORE the final demo to schedule a make-up.

<u>Grading Scale</u>

Grade	Average
A	93 or greater
A-	90 - 92
B+	87 - 89
B	83 - 86
B-	80 - 82
C+	77 - 79
C	73 - 76*
C-	70 - 72
D+	67 - 69
D	63 - 66
D-	60 - 62
F	less than 60

* REMINDER: A passing grade for core classes in CS is a C or above. A C-, 72 or below, is not a passing grade for CS/ECE majors.

Academic Dishonesty

I encourage students to work together and learn from one another on labs and assignments. However, I do expect you to turn in your OWN work for every assignment. Assignments are NOT paired-programming, and all assignments are checked for similarities with others in the class, prior class assignments, and work published online!!! Working with someone does not include copying someone else's work and changing a small amount of that work, such as variable names, comments, spacing, etc. During group assignments you and your partners may turn in one assignment per group with everyone's name attached. Working together is discouraged on exams and the final. At NO point should you copy work from the internet, and if you do copy material from an external resource, then you need to cite the resource and author(s). Paying someone to complete you work is unacceptable and will result in immediate referral to the university!!! Cheating and plagiarism are not taken lightly!

You will receive a zero on your first abuse of these rules, and in the case of shared work, the student sharing the work and the student copying the work will both receive zeros. In addition, the academic dishonesty charge will be documented and sent to your school's dean and the Office of Student Conduct. The bottom line is: Each student is expected to understand all aspects of the programs s/he submits for credit!!!

Please, read the university dishonesty policy:

OAR 576-015-0020 (2) Academic or Scholarly Dishonesty

Students with Disabilities

"Accommodations for students with disabilities are determined and approved by Disability Access Services (DAS). If you, as a student, believe you are eligible for accommodations but have not obtained approval please contact DAS immediately at 541-737-4098 or at http://ds.oregonstate.edu. DAS notifies students and faculty members of approved academic accommodations and coordinates implementation of those accommodations. While not required, students and faculty members are encouraged to discuss details of the implementation of individual accommodations."

Students with documented disabilities who may need accommodations, who have any emergency medical information the instructor should be aware of, or who need special arrangements in the event of evacuation, should make an appointment with the instructor as early as possible, and no later than the first week of the term. Class materials will be made available in an accessible format upon request.

Religious Accommodation of Students Policy

Oregon State University recognizes a diverse group of students, and the university accommodates diverse religious holidays. Please read the policy on religious accommodations for students: religious_accomodations

Appendix B:

Below is an example of an Assignment document students would receive in CS 161 Winter 2019 class. Notice that it does not tell the students how to structure their solution but merely provides the requirements for the program to work.

Assignment #5: 1D Array and Recursion (150 pts)

Design Due: Sunday, 2/24/19, 11:59pm

Make sure you demo Assignment #4 within two weeks of the due date to receive full credit. If you go outside the two-week limit without permission, you will lose 50 points. If you fail to show up for your demo without informing anyone, then you will automatically lose 10 points.

(10 pts) Assignment 4 Reflective Post-Peer Review Survey:

http://oregonstate.qualtrics.com/jfe/form/SV_5b9JiqnokplOIuN

Part 1 and Part 2 details (please refer to the assignment on the class website).

This assignment consists of two mini-programs, word_frequency.cpp and fractals.cpp. You will write a design and code for both. The design can be in one document, but the code will be separate programs.

(50 pts) Design Document – Due Sunday 2/24/19, 11:59pm

Refer to the Example Polya Document - Polya_template.pdf

You need to provide a design for both parts/programs!

Step 1: Understanding the Problem/Problem Analysis. (15 pts)

Do you understand everything in the problem? List anything you do not fully understand, and make sure to ask a TA or instructor about anything you do not understand.

- What are the user inputs/requirements, program outputs, etc.? (5 pts)
- What assumptions are you are making? (5 pts)
- What are all the tasks and subtasks in this problem? (5 pts)

Step 2: Program Design. (25 pts)

- What does the overall big picture of this program look like? (flowchart or pseudocode) (20 pts)
    - What data do you need to create, when do you read input from the user?
    - What are the decisions that need to be made in this program?
    - What tasks are repeated?
    - How would you modularize the program, how many functions are you going to create, and what are they?

- What kind of bad input are you going to handle? (5 pts)

Based on your answers above, list the **specific steps or provide a flowchart** of what is needed to create. Be very explicit!!!

Step 4: Program Testing. (10 pts)

Create a test plan with the test cases (bad, good, and edge cases). What do you hope to be the expected results?

- What are the good, bad, and edge cases for ALL input in the program? Make sure to provide enough of each and for all different inputs you get from the user.

Electronically submit your Design Document by the design due date and your C++ programs (zip two .cpp files together) by the code due date using Peerceptiv.

Design Peer Reviews Due: Thursday, 2/28/19, 11:59pm

Code and Design Back Evaluations: Sunday, 3/3/19, 11:59pm

This assignment consists of two mini-programs, word_frequency.cpp and fractals.cpp. You will need to zip both files and upload a zipped file with both programs to Peerceptiv. If you are doing the extra credit of keeping a log of pointer errors, then you will also include this in the zip. (Please refer to the class website for assingment details).

(80 pts) Implementation Requirements:

- Must produce two working programs that follows each rule stated above
- Your user interface must provide clear instructions for the user and information about the data being presented
- Use of one-dimensional array/C-style strings is required
- Use of references or pointers is required
- Your program should be properly decomposed into tasks and subtasks using functions. To help you with this, use the following:
    - Make each function do one thing and one thing only
    - No more than 15-20 lines inside the curly braces of any function, including main(). Whitespace, variable declarations, single curly braces, vertical spacing, comments, and function headers do not count.
    - Functions over 20 lines need justification in comments
    - Do not put multiple statements into one line
- No global variables allowed (those declared outside of many or any other function), global constants are allowed
- goto is NOT allowed
- You must not have any memory leaks!

- Make sure you follow the style guidelines, have a program and function headers with appropriate comments, and be consistent

(10 pts) Extra Credit:

Keep a journal of errors you run into using pointers in this assignment. You must record 1) whether the error was at compile time or runtime, 2) what the error was, 3) the line number it occurred on, 4) what you did to fix it, and 5) why this fixed it.

(10 pts) Assignment 5 Predictive Pre-Peer Review Survey:

http://oregonstate.qualtrics.com/jfe/form/SV_aeH9GJKWISJJu2F

Appendix C

Rubrics for all five peer reviews for coding solution.

Assignment 1

Following Instructions

Please provide additional comments on anything else that has to do with following assignment instructions.

- Include climits
  - The student included the climits library and prints all 9 short, long, int max and mins.
- Calculate values
  - The student uses pow() and sizeof() to calculate the 9 short, long, and int max and mins.
- Output Labeled
  - The output is properly labeled, i.e. someone could understand what the values mean and which values come from climits and which are from calculations.

Code Readability

Please provide additional comments about the readability of the code.

- Indentation and Spacing
  - There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.
- Program Header
  - There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.
- Comments
  - The program contains enough comments to make the program understandable without knowing the code.

Assignment 2

Following Instructions

You will need to upload the student's code to the server and run their code to knwo whether the student folled instructions. Please provide any comments about the requirements for the operation of the program, such as the choices must be based on numbers, you must handle an invalid number choice, and are there lines separating the navigation paths.

- line separating each navigation
  - Is there a line separating each navigation and the choices are based on numbers?
- 2 paths with a depth of 3, and decisions with 2 choices
  - Are there at least 2 paths that have a depth of 3 (an if with an if that has an if), and do all decisions have 2 choices?
- Handle bad input
  - Does the user make sure there is an else on all decisions that handle bad number input for a choice?
- Create a text adventure game with element of chance
  - Is there an element of chance in the text adventure game that is implemented with rand?

Code Readability

Please provide additional comments about the readability of the code.

- Indentation and Spacing
  - There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.
- Program Header
  - There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.
- Comments
  - The program contains enough comments to make the program understandable without knowing the code.

Assignment 3

Following Instructions

You may need to upload the student's code to the server and run their code to know whether the student followed instructions. Please provide any comments about the requirements for the operation of the program, such as the providing clear instructions for playing the game, making sure players can put money in their bank and bet before each round, etc.line separating each navigation

- Program Interface
  - The program has a clear interface and instructions for how to interact with the game.
- Two functions and global variables
  - Does the program contain at least two functions and no global variables?
- Handle bad input
  - Does the program at least make sure to handle errors within the same type, such as catching a bet more than what a player has in the bank, putting negative values in the bank or bet, etc.

Code Readability

Please provide additional comments about the readability of the code.

- Indentation and Spacing
  - There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.
- Program Header
  - There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.
- Comments
  - The program contains enough comments to make the program understandable without knowing the code.
- Variables
  - Variables are clearly named to make it understandable what information they hold.
- Appropriate Loops
  - The loops used are the appropriate loops for the job they are performing.

Assignment 4

Code Readability

Please provide additional comments about the readability of the code.

- Indentation and Spacing
    - There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.
- Program Header
    - There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.
- Comments
    - The program contains enough comments to make the program understandable without knowing the code.
- Variables
    - Variables are clearly named to make it understandable what information they hold.
- Appropriate Loops
    - The loops used are the appropriate loops for the job they are performing.
- Functions
    - All the functions are small (under 15-20 lines) and do one thing and one thing only.
- Use of references/pointers and global variables
    - Does the program contain at one use of references or pointers and no global variables?

Assignment 5

Code Readability

Please provide additional comments about the readability of the code.

- Indentation and Spacing
  - There is consistent (always use the same thing, 2 spaces, 3 spaces, tab, etc.) and proper (blocks of code that are a part of a specific construct are indented to show relationships) horizontal spacing, as well as vertical spacing between common themes within the program.
- Program Header
  - There is a program header with the name of the program, date created, and good detail in the description of the program with input and output information appropriate for the problem.
- Comments
  - The program contains enough comments to make the program understandable without knowing the code.
- Variables
  - Variables are clearly named to make it understandable what information they hold.
- Appropriate Loops
  - The loops used are the appropriate loops for the job they are performing.
- Functions
  - All the functions are small (under 15-20 lines) and do one thing and one thing only.
- Use of arrays/c-strings and global variables
  - Does the program contain a c-string for the sentence, a dynamic array of C++ or C-style strings, and no global variables?

Appendix D

The following pages are examples of peer reviews. Details of each one can be found below.

Example 1:

Review Score details:

- Include climits: 7/7
- Calculate Values: 7/7
- Output Labeled: 7/7
- Indentation and Spacing: 7/7
- Program Header: 7/7
- Comments: 5/7

Comments:

Your code is readable even for a novice like me. Some areas of improvement I would say are your comments and line character length. Be careful of making a single line too long it will wrap on some terminals. A few of yours did on mine but it was easily correctable by making the terminal window full screen. As far as comments go, I would say you could use more. This assignment was simple, but comments will play a big role on more complicated programs. Also it looks like you copy and pasted the second block comment and said calculates preprocessor marcos which I am not sure is correct. I would say that the second section contains functions used to calculate number type size as opposed to macros like the first section did but you could know more than me. Just some food for thought. Nice work.

Back Evaluation:

Score: 5/5

Comments:

Thanks for your review. I will try to reduce the length of my lines for future assignments. Also, I will include more comments. It was hard to figure out exactly what to comment on with an assignment. Thanks for your review it is the best I have received!

Example 2:

Review Score details:

- line separating each navigation: 7/7

- 2 paths with a depth of 3, and decisions with 2 choices: 7/7
- Handle bad input: 7/7
- Create a text adventure game with element of chance: 7/7
- Indentation and Spacing: 7/7
- Program Header: 7/7
- Comments: 7/7

Comments:

It would be very easy to read your code if I was more skilled at coding! I am a first-time coder, so it took some time to try and understand how you did things. I was happy to learn a multitude of things from your program. I thought it was cool how you used the "cin.ignore" to enable the user to hit Enter to continue in the beginning. I also thought it was cool that you incorporated the random numbers without having to prompt the user, it was very subtle how the story would change direction with each time you played based on the rand. Very cool! The actual code was very well spaced and easy to understand. The only mistakes I could find was I think you should add a 'n\' on line 229 after the word "heard" , so in the executed program it will do a line break rather than a continuous run on. That was the only thing I could find! Also you misspelled "convenience" on line 255...I have to critique you somehow, since your code was perfect and I have nothing bad to say about it! Good job :)

Back Evaluation:

Score: 5/5

Comments:

I'm so happy that I was actually able to help a fellow programmer to learn from what I've done. I appreciate the little notes that you point out as well. Without critique, even in the smallest of facets, how are we expected to improve! I'll take into account your comments for future assignments that I do.

Example 3:

Review Score details:

- Problem-Solving Steps 1, 2, 4: 7/7
- Understanding the Problem: 7/7
- Plan Quality: 7/7
- Plan/Algorithm: 7/7
- Program Testing: 5/7

Comments:

You obviously have a solid grasp of the problem, and described everything well, including bringing up a couple of things I hadn't even thought of yet, so I'm not grading you down for possibly not explicitly and separately stating a couple of the bullet points in the Program Design section. It's already messed up that I get to weigh in on your grade, especially when a few of the bullet points are repetitive anyway and the point of design is to be useful, not to make busy work. Your design is definitely useful.

If you care, here are a few nitpicky things that could make your doc easier to skim and make sure you get full credit from other reviewers in the future. (If not, skip the rest and go about your day!)

1. Your paragraph restating the problem seems solid, you clearly read and understand the assignment. Don't be afraid to split this into a few smaller paragraphs, otherwise it's just a lot of info in one block.

2. Your misunderstandings section shows that you are actually thinking about how to make the game as realistic as possible. I don't think either of those things are requirements of the program, but hey, if you manage to program in those aspects (Ace being a 10 vs. 1, commonness of certain cards), I will be impressed and your game will be cooler. Either way, it showed me that you actually thinking about ways to optimize the assignment.

3. Your design steps are thorough and adequately describe what should happen, and bad input handling. It is a bit hard to get a visual sense of how things work from a list, so it might have been easier to read if supplemented with a simple flowchart or some pseudocode. This might be a personal thing because of my background in graphic design, so consider my own bias here and take with a grain of salt.

4. It would have made it easier to check for requirements if you put Bad Input Handling in its own section, since the assignment has this as a separate 5-point thing.

5. Let me know what you consider to be good/bad/edge cases in your testing table. You obviously hit all the marks, and I realize that edge cases are weird, but throw on a label so I can see that you have considered this aspect of the assignment.

If any of the above isn't helpful, ditch it and do what works for you! Great job. :)

Back Evaluation:

Score: 3/5

Comments:

The first review that has seemed genuine and that I could actually take advice from. I will seriously consider doing a flowchart to make it easier to follow. I just don't like doing them as much because I like to just type on my computer and such. Keep up the good work on reviewing, this was the best review by a lot. And I'm sure more people will use most of the information, so it is very helpful. :)

Example 4:

Review Score details:

- Problem-Solving Steps 1, 2, 4: 7/7
- Understanding the Problem: 7/7
- Plan Quality: 7/7
- Plan/Algorithm: 7/7
- Program Testing: 7/7

Comments:

Overall, the design is in great detail. The only thing I can really comment on is that your flow chart does not use the standard flowchart shape meanings. It's nothing too big as I can still understand what the flow chart is doing, but flow charts have their own language-- i.e. diamonds are usually used for a choice in the program, squares are a process, rhombuses are inputs/outputs. Again, it's nothing too big, and it really only matters when you are actually in the professional world. Besides that, your design document was very well done and in great detail, keep up the good work.

Back Evaluation:

Score: 3/5

Comments:

It doesn't matter in my opinion, this is not a company that requires perfection in flowcharts. It only matters that the assignment is understood and what is necessary for the program.