CPSC-354 Report

John Robert Mulhern
Chapman University

September 1, 2024

Abstract

This document will contain the various assignments completed by John Mulhern over the course of the CPSC 354 Programming Languages course. For any questions, comments, or concerns in this document, feel free to reach out to John Mulhern at his email, mulhern@chapman.edu, or by phone number: (208)-451-3484.

Contents

1 Introduction

This report will document my learning throughout the course. It will be a collection of my notes, homework solutions, and critical reflections on the content of the course. Something in between a semester-long take home exam and my own lecture notes.[1]

To modify this template I would need to modify the source `report.tex` which is available in the course repo. For guidance on how to do this read both the source and the pdf of `latex-example.tex` which is also available in the repo. Also check out the usual resources (Google, Stackoverflow, LLM, etc). It was never as easy as now to learn a new programming lanugage (which, btw, LaTeX is).

For writing LaTeX with VSCode use the LaTeX Workshop extension.

There will be deadlines during the semester, graded mostly for completeness. That means that I will get the points if I submit in time and are on the right track, independently of whether the solutions are technically correct. I will have the opportunity to revise my work for the final submission of the full report.

[1] One purpose of giving the report the form of lecture notes is that self-explanation is a technique proven to help with learning, see Chapter 6 of Craig Barton, How I Wish I'd Taught Maths, and references therein. In fact, the report can lead you from self-explanation (which is what you do for the weekly deadline) to explaining to others (which is what you do for the final submission). Another purpose is to help those of you who want to go on to graduate school to develop some basic writing skills. A report that you could proudly add to your application to graduate school (or a job application in industry) would give you full points.

The full report is due at the end of the finals week. It will be graded according to the following guidelines.

Grading guidelines (see also below):

- Is typesetting and layout professional?
- Is the technical content, in particular the homework, correct?
- Did I find interesting references [BLA] and cites them throughout the report?
- Do the notes reflect understanding and critical thinking?
- Does the report contain material related to but going beyond what we do in class?
- Are the questions interesting?

Do not change the template (fontsize, width of margin, spacing of lines, etc) without asking your first.

2 Week by Week

2.1 Week 1

Tuesday: Orientation and introduction to the course.
Thursday: First Lab on Tuesday's content.

Notes and Homework

Our homework for this week was to finish levels 5-8 of the tutorial world inside the Natural Numbers Game provided to us in class. On Tuesday of week 1, we went over the game in basic detail, covering levels 1-4 as to become used to the website so we could begin our first challenge. The solutions to each of the worlds can be found below in an itemized format.

- World 5 Solution: rw[add _zero] , rw[add _zero] , rfl.
- World 6 Solution: rw[add _zero c] , rw[add _zero b] , rfl.
- World 7 Solution: rw[one _eq _succ _zero] , rw[add _succ] , rw[add _zero] , rfl
- World 8 Solution: rw[two _eq _succ _one, one _eq _succ _zero] , rw[add _succ] , rw[add _zero] , rw[<- one _eq _succ _zero, <- two _eq _succ _one, <- three _eq _succ _two, <- four _eq _succ _three]

World 5's solutition is to rewrite the equation by adding zero onto a variable **b** or **c**, and as we know from a descrete math proof described as $\forall n \in N, n+0 = n$, adding zero to any number provides the same equivalent number. This becomes useful later in sections seven and eight when we begin dealing with adding zero to successor values.

Comments and Questions

Looking at Discrete Math over the past few days and getting a refresher on the course since I took it a few semesters ago has been very interesting. I have enjoyed the tutorial levels of the Natural Numbers Game, and I particularly enjoyed their explanations for the proofs and early concepts for Discrete Math. Had I known about this website when I was taking the course, I have a feeling it would have been a great resource to support my understanding of those proofs and other concepts.

My question then in relation to discrete mathematics comes more so with how we utilize those proofs on paper verses when they are used in a computational environment. For example, writing a Discrete Math proof can often take a significant amount of time and paper to create, depending of course on the operation. For something as simple as addition or multiplication, using the proofs outlined in Discrete Math can turn a

simple problem, such as 2*(3+2+4), into a massive multi-page proof. However, when a computer runs such a problem, it concludes the correct answer in an astoundingly short amount of time.

My question is then, what is the largest, hardest, and most difficult proof a person could do by hand that can be done in seconds by a machine?[2]

2.2 Week 2

(Delete:) Week 2 (and all the other weeks) should follow the same pattern as Week 1. Even if there is a week without homework, notes and comments (see above) are still expected.

2.3 ...

...

3 Lessons from the Assignments

(Delete and Replace): Write three pages about your individual contributions to the project.

On 3 pages you describe lessons you learned from the project. Be as technical and detailed as possible. Particularly valuable are *interesting* examples where you connect concrete technical details with *interesting* general observations or where the theory discussed in the lectures helped with the design or implementation of the project.

Write this section during the semester. This is approximately a quarter of apage per week and the material should come from the work you do anyway. Just keep your eyes open for interesting lessons.

Make sure that you use LaTeX to structure your writing (eg by using subsections).

4 Conclusion

(Delete and Replace): (approx 400 words) A critical reflection on the content of the course. Step back from the technical details. How does the course fit into the wider world of software engineering? What did you find most interesting or useful? What improvements would you suggest?

References

[BLA] Author, Title, Publisher, Year.

[2] It is important to learn to ask *interesting* questions. There is no precise way of defining what is meant by interesting. You can only learn this by doing. An interesting question comes typically in two parts. Part 1 (one or two sentences) sets the scene. Part 2 (one or two sentences) asks the question. A good question strikes the right balance between being specific and technical on the one hand and open ended on the other hand. A question that can be answered with yes/no is not an interesing question.