

# CPSC-406 Report

Charlie Conner  
Chapman University

February 15, 2026

## Abstract

This is the place to write an abstract. The abstract should be a short summary of the report. It should be written in a way that makes it possible to understand the purpose of the report without reading it. You can write a dummy abstract first and replace it with a real one later in the semester.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Week by Week</b>	<b>2</b>
2.1	Week 0 (EXAMPLE) . . . . .	2
2.1.1	Notes . . . . .	2
2.1.2	Homework . . . . .	2
2.1.3	Exploration . . . . .	3
2.1.4	Questions . . . . .	3
2.2	Week 1 . . . . .	4
2.2.1	Notes . . . . .	4
2.2.2	Homework . . . . .	4
2.2.3	Exploration . . . . .	4
2.2.4	Questions . . . . .	4
<b>3</b>	<b>Synthesis</b>	<b>4</b>
<b>4</b>	<b>Evidence of Participation</b>	<b>5</b>
<b>5</b>	<b>Conclusion</b>	<b>5</b>

## 1 Introduction

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

---

<sup>1</sup>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.

To write your own report, you start from `report.tex` which is available in the course repo. For guidance on how to do this read on and also consult `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 language (which, btw, L<sup>A</sup>T<sub>E</sub>X is).

For writing L<sup>A</sup>T<sub>E</sub>X with VSCode, consider using the [L<sup>A</sup>T<sub>E</sub>X Workshop](#) extension.

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

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 the student find interesting references [BLA] and cite 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 0 (EXAMPLE)

Week 1 aligns with the first week of the semester.

If you think that the writing flows better if you merge the sections “Notes” and “Homework”, you can do so, but keep the heading for “Comments and Questions”.

#### 2.1.1 Notes

This section is optional.

Our experience is that writing notes is a great way to learn. You can use this section to write your own notes and showcase your own understanding.

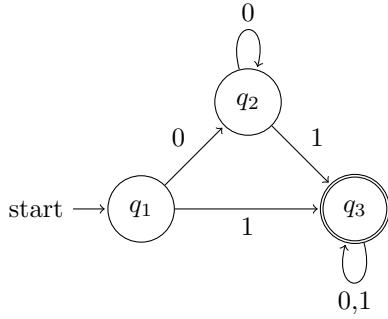
#### 2.1.2 Homework

This section will typically contain Homework problems. You should write up your solutions in L<sup>A</sup>T<sub>E</sub>X. You can use the `lstlisting` environment to include code. You can use [Excalidraw](#) for drawings. Pictures from handwritten drawings are acceptable if the drawings are of high quality (pictures from rough notes and quick sketches are likely to lose you points).

Make sure that this section can be read without referring back to the homework question. Introduce the question/problem and repeat it in your own words. Make sure to typeset your homework in a way that makes it clear what the question and what the answer is. Present it as a worked example would be presented in a textbook.

Also explain what you learn from the homework. Each homework was carefully drafted to bring home a particular teaching point. Make sure to explain what this point is. Relate it to the big questions mentioned above.

In case you want to draw automata in L<sup>A</sup>T<sub>E</sub>X, you can use the tikz package. Here is an example of a simple automaton:



By the way, ChatGPT is quite good at outputting tikz code. Another alternative package is xymatrix.

### 2.1.3 Exploration

Here are some hints. Why is this material included in the course? What are the big questions that motivate the study of this subject? How does this material connect to broader themes or issues in the field? What practical or theoretical problems can be addressed through an understanding of these topics? A great way to test whether you understand the material is to make your own exercises and answer them. Material related to but going beyond what we do in class is welcome.

Feel free to use your favourite LLM to help you build a mental landscape of the subject. Think of LLMs as an extension of Wikipedia and Google, a tool you should be using as introduction to any subject. If you didn't check with Google, Wikipedia and GPT, you are not ready to write your own notes. On the other hand, while using these resources is necessary, you need to always exercise your own critical thinking and you are always responsible for what you write.

### 2.1.4 Questions

Ask at least one **interesting question**<sup>2</sup> on the lecture notes. Also post the question on the Discord channel so that everybody can see and discuss the questions.

---

<sup>2</sup>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.

## 2.2 Week 1

### 2.2.1 Notes

This section is optional.

### 2.2.2 Homework

### 2.2.3 Exploration

### 2.2.4 Questions

## 3 Synthesis

(approx 1 page, plus references)Section 2 gives you an opportunity to practice "skill drill" and to explore the material in more depth. The purpose of this section is to synthesize the knowledge you gained. Since Programming Languages is a wide field, it may be appropriate to focus on a particular topic of your choice.

We suggest the following timeline. Week 5-8: Decide on a topic and discuss it with your instructor in the office hours. Write a summary email to your instructor after office hours, including the feedback you got with further reflection and planning. Week 9-12: Write a draft of your synthesis and discuss it with your instructor during office hours. Again, write a summary email to your instructor after office hours, including the feedback you got with further reflection and planning. The final version of the Synthesis is due with the rest of the final report.

Here are some example titles you can think of for this section:

- **Standard Academic Titles**

- "Synthesis and Reflection"
- "Integration of Concepts"
- "Synthesis of Learning"
- "Critical Synthesis"

- **Algorithm Analysis Focus**

- "The Core of the Code: A Personal Synthesis"
- "Connecting the Dots: From Theory to Practice"
- "The Big Picture: Algorithm Analysis in Context"
- "Principles of Problem-Solving"

- **Concept-Focused Titles**

- "Beyond the Code: Understanding the Essence of Algorithms"
- "How long is too long? A Case Study in Complexity Theory"
- "Principles of Computation"
- "Abstract Machines for Concrete Problems"
- "Is there an Algorithm for this? On (Un)Decidability"

- **Process-Focused Titles**

- "My Journey Through Algorithm Analysis"
- "From Implementation to Understanding"

- “Building Mental Models of Practical Problems”
- “Discovering the Foundations behind Algorithms”

## 4 Evidence of Participation

## 5 Conclusion

(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.