

# CPSC-354 Report

Keoni Lanoza  
Chapman University

September 11, 2022

## Abstract

Short summary of purpose and content.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	General Remarks . . . . .	1
1.2	LaTeX Resources . . . . .	2
1.2.1	Subsubsections . . . . .	2
1.2.2	Itemize and enumerate . . . . .	2
1.2.3	Typesetting Code . . . . .	2
1.2.4	More Mathematics . . . . .	2
1.2.5	Definitons, Examples, Theorems, Etc . . . . .	3
1.3	Plagiarism . . . . .	3
<b>2</b>	<b>Homework</b>	<b>3</b>
2.1	Week 1 . . . . .	3
2.2	Week 2 . . . . .	4
2.3	Week 3 . . . . .	4
<b>3</b>	<b>Project</b>	<b>4</b>
3.1	Specification . . . . .	5
3.2	Prototype . . . . .	5
3.3	Documentation . . . . .	5
3.4	Critical Appraisal . . . . .	5
<b>4</b>	<b>Conclusions</b>	<b>5</b>

## 1 Introduction

Replace this entire Section 1 with your own short introduction.

### 1.1 General Remarks

First you need to [download and install](#) LaTeX.<sup>1</sup> For quick experimentation, you can use an online editor such as [Overleaf](#). But to grade the report I will used the time-stamped pdf-files in your git repository.

---

<sup>1</sup>Links are typeset in blue, but you can change the layout and color of the links if you locate the `hypersetup` command.

LaTeX is a markup language (as is, for example, HTML). The source code is in a `.tex` file and needs to be compiled for viewing, usually to `.pdf`.

If you want to change the default layout, you need to type commands. For example, `\medskip` inserts a medium vertical space and `\noindent` starts a paragraph without indentation.

Mathematics is typeset between double dollars, for example

$$x + y = y + x.$$

## 1.2 LaTeX Resources

I start a new subsection, so that you can see how it appears in the table of contents.

### 1.2.1 Subsubsections

Sometimes it is good to have subsubsections.

### 1.2.2 Itemize and enumerate

- This is how you itemize in LaTeX.
- I think a good way to learn LaTeX is by starting from this template file and build it up step by step. Often stackoverflow will answer your questions. But here are a few resources:
  1. [Learn LaTeX in 30 minutes](#)
  2. [LaTeX – A document preparation system](#)

### 1.2.3 Typesetting Code

A typical project will involve code. For the example below I took the LaTeX code from [stackoverflow](#) and the Haskell code from [my tutorial](#).

---

```
-- run the transition function on a word and a state
run :: (State -> Char -> State) -> State -> [Char] -> State
run delta q [] = q
run delta q (c:cs) = run delta (delta q c) cs
```

---

Short snippets such as `run :: (State -> Char -> State) -> State -> [Char] -> State` can also be directly fitted into text. There are several ways of doing this, for example, `run :: (State -> Char -> State) -> State ->` is slightly different in terms of spaces and linebreaking (and can lead to layout that is better avoided), as is

```
run :: (State -> Char -> State) -> State -> [Char] -> State
```

For more on the topic see [Code-Presentations Example](#).

Generally speaking, the methods for displaying code discussed above work well only for short listings of code. For entire programs, it is better to have external links to, for example, Github or [Replit](#) (click on the "Run" button and/or the "Code" tab).

### 1.2.4 More Mathematics

We have already seen  $x + y = y + x$  as an example of inline maths. We can also typeset mathematics in display mode, for example

$$\frac{x}{y} = \frac{xy}{y^2},$$

Here is an example of equational reasoning that spans several lines:

$$\begin{array}{ll} \text{fib}(3) = \text{fib}(1) + \text{fib}(2) & \text{fib}(n+2) = \text{fib}(n) + \text{fib}(n+1) \\ = \text{fib}(1) + \text{fib}(0) + \text{fib}(1) & \text{fib}(n+2) = \text{fib}(n) + \text{fib}(n+1) \\ = 1 + 0 + 1 & \text{fib}(0) = 0, \text{fib}(1) = 1 \\ = 2 & \text{arithmetic} \end{array}$$

### 1.2.5 Definitions, Examples, Theorems, Etc

**Definition 1.1.** This is a definition.

**Example 1.2.** This is an example.

**Proposition 1.3.** *This is a proposition.*

**Theorem 1.4.** *This is a theorem.*

You can also create your own environment, eg if you want to have Question, Notation, Conjecture, etc.

## 1.3 Plagiarism

To avoid plagiarism, make sure that in addition to [PL] you also cite all the external sources you use. Make sure you cite all your references in your text, not only at the end.

## 2 Homework

This section will contain your solutions to homework. For every week, you will have a subsection that contains your answers.

### 2.1 Week 1

```
print("Please enter integer A")
aString = input("a: ")
while True:
    try:
        a = int(aString)
        break
    except ValueError:
        print("Not an integer. Please try again.")
        print("Please enter INTEGER A")
        aString = input("a: ")

print("Please enter integer B")
bString = input("b: ")
while True:
    try:
        b = int(bString)
        break
    except ValueError:
        print("Not an integer. Please try again.")
        print("Please enter INTEGER B")
        bString = input("b: ")
```

```

a = int(aString)
b = int(bString)

while (a != b):
    if (a > b):
        a = a - b
    elif (b > a):
        b = b - a

print ("The greatest common divisor is: " + str(a))

```

This program works by first asking the user to input an integer named "A". The program stores this input in a variable called "aString" and then utilizes error checking to make sure the user's input can be converted into an integer. If the input cannot be converted into an integer, the program loops until the user enters valid input. If the user's input passes error checking, the program then prompts the user for an integer named "B" and follows the same error-checking process. Once the user passes error checking for both variables, the program enters a loop. In the loop, if integer A is greater than integer B, then integer A is replaced with the value of integer A - integer B. If integer B is greater than integer A, then integer B is replaced with the value of integer B - integer A. This process repeats until integer A is equal to integer B. When this is reached, the greatest common divisor of the original integer A and original integer B is printed out to the user.

For example, if we took A to be an integer representing the value of 9, and B to be an integer representing the value of 33, the program would follow this process: Because B, which equals 33 is greater than A, which equals 9. B's value would be replaced with B - A, which is 24. B is still greater than A, so B's value would be replaced by B - A again which now equals 15. 15 is still greater than 9, so B would become 6. Now, A with a value of 9 is greater than B which has a value of 6. A would be replaced with A - B which equals 3. This makes B greater than A again. B is now replaced with B - A, or 6 - 3, which equals 3. Now that A and B are equal, the greatest common divisor, which is 3 because both A and B equal 3, is output to the user.

## 2.2 Week 2

The example I will be choosing from Homework 2 is the less\_equal function I implemented in Haskell with the following lists [1,2,3] and [2,3,4]. My code is uploaded in Homework2.hs. It works by taking in two lists (x:xs) and (y:ys) as input. It functions recursively, by comparing the heads (x and y) of each list and returning a value of False whenever x is greater than or equal to the value of y. If in the current iteration it is less than or equal to the value of y, then it proceeds via recursion to evaluate the rest of the lists (xs and ys). If it is able to get to the end of the list, then it returns a value of True.

For the two example lists [1,2,3] and [2,3,4], because each value in the first list is less than the corresponding value in the second list (by index), the program returns True.

## 2.3 Week 3

...

## 3 Project

Introductory remarks ...

The following structure should be suitable for most practical projects.

### **3.1 Specification**

### **3.2 Prototype**

### **3.3 Documentation**

### **3.4 Critical Appraisal**

...

## **4 Conclusions**

(approx 400 words)

In the conclusion, I want 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 programming languages and software engineering?

## **References**

[PL] [Programming Languages 2022](#), Chapman University, 2022.