

CPSC-354 Report

Michael Masakayan
Chapman University

September 5, 2022

Abstract

Short summary of purpose and content.

Contents

1	Introduction	1
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
2	Homework	3
2.1	Week 1	3
2.2	Week 2	4
3	Project	4
3.1	Specification	4
3.2	Prototype	4
3.3	Documentation	4
3.4	Critical Appraisal	4
4	Conclusions	4

1 Introduction

¹ My name is Michael Masakayan, I'm a 4th year computer science major. I transferred from Pasadena City College in 2020. I am interested in cyber security and although I won't be able to get a minor in it I plan on getting a few certificates this year. I enjoy playing video games and like watching movies

1.1 General Remarks

First you need to [download and install](#) LaTeX.¹ 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.

¹Links 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.

2.1 Week 1

Assignment 1: GCD. The program figures out the Greatest Common Divisor for any given two numbers using Euclid's algorithm.

1. Firstly you want to put the program into your IDE and run `g++ -o assignment1 assignment.cpp`
2. then type in `./assignment1.cpp`
3. Once you run the program you type in the first number then you hit enter then you type in the second number and hit enter. It will then show you the GCD of the two numbers.

I chose to use c++ because I am familiar with the language and it was easier for me to write the program in it. This assignment was mainly for us to get used to using LaTeX

```
#include <iostream>
using namespace std;
int gcd(int n, int m) {
    if (m == 0)
    {
        return n;
    }

    else if(n>m)
    {
        gcd(m,n-m);
    }
}
```

```

    else if(n<m)
    {
        gcd(n,m-n);
    }
    return gcd(m, n % m);
}
int main() {
    int n,m;
    cout<<"type the number for the first argument of the GCD, then hit enter"<<'\\n';
    cin>>n;
    cout<<"type the number for the first argument of the GCD,then hit enter"<<'\\n';
    cin>>m;
    cout<<"GCD of "<< n <<" and "<< m <<" is "<< gcd(n, m);
    return 0;
}

```

2.2 Week 2

...

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.