CPSC-354 Report

Stephanie Munday
Chapman University

September 11, 2022

Abstract

Short summary of purpose and content.

Contents

1 Introduction

Replace this entire Section 1 with your own short introduction.

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.

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`.

[1] Links are typeset in blue, but you can change the layout and color of the links if you locate the `hypersetup` command.

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:

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

1.2.5 Definitons, 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

HW 1 - Greatest Common Divisor

```
def gcd(n, m):
    while n != m:
        if  n > m:
            n = n-m
        else:
            m = m-n
    return n
```

The code above implements Euclid's algorithm to find the greatest common divisor in python. Below is an explanation given sample input gcd(9,33).

While n != m, the code will compare whether or not n is greater than m. If n > m, n will become n – m. Otherwise if n < m, m will become m – n. When n == m, the greatest common divisor has been found.

Keeping this logic in mind, let n = 9, m = 33.

gcd(9,33) =

gcd(9,24) =

gcd(9,15) =

gcd(9,6) =

gcd(3,6) =

gcd(3,3) =

3

Since n == m and the value of both is 3, the greatest common divisor is 3 for this example.

2.2 Week 2

HW 2 - Recursion in Functional Programming

```
select_evens :: [a] -> [a]
select_evens [] = []
select_evens (x:(y:xs)) = y:select_evens(xs)

select_odds :: [a] -> [a]
select_odds [] = []
select_odds (x:(y:xs)) = x:select_odds(xs)

member :: (Eq a) => a -> [a] -> Bool
member a [] = False
member a (x:xs)
    | a == x = True
    | otherwise = a `member` xs

append :: (Ord a) => [a] -> [a] -> [a]
append [] [] = []
append [] ys = ys
append (x:xs) (ys) = x:append(xs) (ys)


revert :: [a] -> [a]
revert [] = []
revert (x:xs) = revert xs ++ [x]

less_equal :: (Ord a) => [a] -> [a] -> Bool
less_equal [] [] = True
less_equal (x:xs) (y:ys)
    | x > y   = False
    | otherwise = xs `less_equal` ys
```

The code above implements select_evens, select_odds, member, append, revert, less_equal as recursive functions in Haskell. Below is an explanation showing computations for append given inputs [1,2] [3,4,5].

append [1,2] [3,4,5]
    append [2] [3,4,5]
        append [] [3,4,5]

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.

[P] Punctuation, StackExchange, 2022.

[S] Spacing, StackExchange, 2022.