CPSC-354 Report

Your Name
Chapman University

September 18, 2022

Abstract

Short summary of purpose and content.

Contents

1 Introduction

Replace this entire Section **??** 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`.

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

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.

```haskell
-- 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:

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

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 [?] 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

I have added my program into this github repo. It is a simple python program that loops infinatly until a and b are equal. I ran it with python 3 and manually input 9 and 33 yeilding the answer 3.

2.2 Week 2

This week we were tasked with creating simple recursive programs in huskall. These are my solutions for all of the assigned functions. These functions can also be found in a huskell file titled "Main.hs".

```haskell
-- Takes a list of char and returns a list of char. This works by assinging the element at odd
    positional values of a list into an empty list using the zip function which is what it
    returns.
select_evens :: [[a]] -> [[a]]
select_evens xs = [x | (x,i) <- zip xs [0..], odd i]
```

I referenced this.

```haskell
-- Same logic as above just assignes based on even possition.
select_odds :: [[a]] -> [[a]]
select_odds xs = [x | (x,i) <- zip xs [0..], even i]
```

```haskell
-- Uses the filter to function to create a list of all matching elements to the ones provided.
    If the length is greater then 0 the its a member.
member :: Int -> [Int] -> IO Bool
member x li= do
  let xs = filter(== x) li
  if length xs == 0
      then
          return (False)
      else do
          return(True)
```

```haskell
-- Uses ++ to concatinate lists.
append :: [Int] -> [Int] -> [Int]
append l1 l2 = l1 ++ l2
```

```haskell
-- Recursively assigns element to back of new list, creating a reverse list.
revert :: [Int] -> [Int]
revert [] = []
revert (x:xs) = revert xs ++ [x]
```

```haskell
-- Compares two strings using <=.
less_equal :: [Int] -> [Int] -> IO Bool
less_equal l1 l2 = do
    if last l1 <= last l2
        then
            return (True)
        else do
            return (False)
```

2.3 Week 3

This week we were tasked with finishing the hanoi file supplied to us. The full file can be found in Hanoi.txt. Hanoi is used 31 times in the file. You can express this as formula because it doesn't matter the starting blocks since the process will always be the exact same. you will always need to reduce to the highest block so the number is nonconsiquental. There are many cool visuals of this online which show what is looks like using a graphical interpritation. There is a simple break down for however many N you have.

...

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.