CPSC-354 Report

Darren Pak
Chapman University

September 12, 2022

Abstract

This is a culmination of all assignments and reports for CPSC-354 taught by Alex Kurz at Chapman University Fall 2022.

Contents

1 Introduction

My name is Darren Pak and I am a computer science major at Chapman University with a minor in Data Analytics. My current goals as of Fall 2022 are to find interesting job opportunities and career paths that I find enjoyable and are able to sustain my lifestyle.

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

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

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:

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

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

In Week 1, I will go over Euclid's Algorithm for Greatest Common Divisor and how it is implemented in C++.

2.1.1 Euclid's Algorithm

Euclid's Algorithm is defined as follows:

gcd(a,b):

Input: Two whole numbers (integers) called a and b, both greater than 0.

(1) if

$$a > b$$

then replace a by a-b and go to (1).

(2) if

$$b > a$$

then replace b by b-a and go to (1).

Output: a

As described in Alex Kurz Homework (Week 1)

2.1.2 Implementation in C++

Below is the example for implementation of Euclid's Algorithm in C++:

```
#include <iostream>

using namespace std;

int gcd(int a, int b) {
    while ((a != 1) && (b != 1)) {
        if (a > b) {
            a = a-b;
        }
        if (b > a) {
            b = b-a;
        }
        if (a == b) {
            return a;
        }
    }
    return a;
}

int main()
{
    cout<<gcd(9,33)<<endl;

    return 0;
}
```

In the function gcd, there is a while loop checking if either of the inputs are 1. This will eliminate cases where the GCD is already the lowest possible and as described in Euclid's algorithm, will return 1. Within the while loop, we go over the two rules in Euclid's Algorithm. The first being if a is greater than b then a is assigned to a - b. The second rule being if b is greater than a then b is assigned to b - a. Next we resolve the output and return a as the result.

2.2 Week 2

Week 2 is focused on recursion and functions in Haskell. For this assignment, I created 6 different functions using recursion in Haskell. Find the full Github Repository here.

2.2.1 Function Select Evens

Here is a code snippet from the above mentioned Github Repository of the Select Evens function.

```
select_evens :: [a] -> [a]
select_evens [] = []
select_evens (x:xs)
    |mod (length xs) 2 == 1 = x : select_evens xs
    |otherwise = select_evens xs
```

This function takes a list as an input and returns a list of only the even index elements. For example, from a list of ["a","b","c","d"] the function would return ["b","d"]. The first line of this function determines the input and outputs which are both lists of elements. The second line determines that an empty list from the function returns an empty list. This will become our indicator for ending recursion. Next we have an if statement saying that after the head, if there are an odd number of elements remaining, then the head element is of an even index. This means it would be appended to the returning list. If there is an even number of elements remaining, this means that the head element is of an odd index, meaning that the head element is skipped and will not be appended to the list. After all of the calculations have completed, the elements are appended to an empty list and added to the front in the order they were calculated.

Collaborated with Adrian Edralin for Week 2 Assignment.

...

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.