

# CPSC-354 Report

Rohm Tandon  
Chapman University

07/01/2024

## Abstract

This report contains assignments throughout the fall 2024 semester and is intended for the purpose of documenting my work and showing my progress in CPSC 354 - Programming Languages, taught by Jonathan Weinberg.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Week by Week</b>	<b>1</b>
2.1	Week 1 . . . . .	1
2.2	Week 2 . . . . .	4
2.3	Week 3 . . . . .	6
<b>3</b>	<b>Lessons from the Assignments</b>	<b>7</b>
<b>4</b>	<b>Conclusion</b>	<b>7</b>

## 1 Introduction

This report, prepared for CPSC 354 - Programming Languages at Chapman University, is a comprehensive account of my academic voyage over the semester. It includes a detailed compilation of my notes, homework solutions, and critical reflections on the coursework. This report serves as a bridge between the theoretical knowledge imparted in lectures and the practical skills essential for future pursuits in both graduate studies and the software industry.

## 2 Week by Week

### 2.1 Week 1

#### Notes

In week 1 we learnt about Lean as a programming language and its correlation to discrete math. We also learnt about other proof assistants. We then shifted our focus to the NNG tutorial world as you can see below.

#### Homework

Tutorial world

Level 5 / 8 : Adding zero

Active Goal

Objects:

$a\ b\ c : \mathbb{N}$

Goal:

$a + (b + 0) + (c + 0) = a + b + c$

rw [add\_zero]

Active Goal

Objects:

$a\ b\ c : \mathbb{N}$

Goal:

$a + b + (c + 0) = a + b + c$

rw [add\_zero]

Active Goal

Objects:

$a\ b\ c : \mathbb{N}$

Goal:

$a + b + c = a + b + c$

rfl

level completed! 🏆

Level 5:

Discrete math's lemmas tell us that anything added to 0 will give the result of that number itself. So;  $A+0=A$ . Using this we can bring the left hand side down to  $a+b+c$ . From here we can use the property of reflexivity to show that both sides are equal, hence solving the puzzle.

## Level 6 / 8 : Precision rewriting

Objects:

$a, b, c : \mathbb{N}$

Goal:

$$a + (b + 0) + (c + 0) = a + b + c$$

```
rw [add_zero c]
```

Active Goal

Objects:

$a, b, c : \mathbb{N}$

Goal:

$$a + (b + 0) + c = a + b + c$$

```
rw [add_zero b]
```

Active Goal

Objects:

$a, b, c : \mathbb{N}$

Goal:

$$a + b + c = a + b + c$$

```
rfl
```

level completed! 🏆

Level 6:

Level 7 / 8 : add\_succ

**Theorem** `succ_eq_add_one` : For all natural numbers  $a$ , we have  $\text{succ}(a) = a + 1$ .

Active Goal

---

**Objects:**  
 $n : \mathbb{N}$

**Goal:**  
 $\text{succ } n = n + 1$

`rw [one_eq_succ_zero]`

Active Goal

---

**Objects:**  
 $n : \mathbb{N}$

**Goal:**  
 $\text{succ } n = n + \text{succ } 0$

`rw [add_succ]`

Active Goal

---

**Objects:**  
 $n : \mathbb{N}$

**Goal:**  
 $\text{succ } n = \text{succ } (n + 0)$

`rw [add_zero]`

Level 7:

Level 8 / 8 : 2+2=4

$2 + 2 = 4.$   
example :  $(2 : \mathbb{N}) + 2 = 4 := \text{by}$ 

```

2 rw[three_eq_succ_two]
3 rw[two_eq_succ_one]
4 rw[one_eq_succ_zero]
5 rw[succ_eq_add_one]
6 rw[one_eq_succ_zero]
7 rw[add_succ]
8 rw[add_zero]
9 rw[add_succ]
10 rw[add_succ]
11 rw[add_zero]
12 rfl

```

Level completed! 🎉

No Goals

Level 8:

## 2.2 Week 2

### Notes

In week 2 we learnt about recursion and its application in other problems such as the Towers of Hanoi game we played. We also learnt about its various benefits such as breaking down complexity of problems and being more concise.

### Homework

Addition world

Level 1 / 5 : zero\_add

**Theorem** zero\_add: For all natural numbers  $n$ , we have  $0 + n = n$ .

```

theorem zero_add (n : ℕ) : 0 + n = n := by
  1 induction n with d hd
  2 rw[add_zero]
  3 rfl
  4 rw[add_succ]
  5 rw[hd]
  6 rfl
  7 |

```

Level 1:

Level completed! 🎉

Level 2 / 5 : succ\_add

**Theorem** succ\_add: For all natural numbers  $a, b$ , we have  $\text{succ}(a) + b = \text{succ}(a + b)$ .

```

theorem succ_add (a b : ℕ) : succ a + b = succ (a + b) := by
  1 induction b with d hd
  2 rw[add_zero]
  3 rw[add_zero]
  4 rfl
  5 rw[add_succ, add_succ]
  6 rw[hd]
  7 rfl
  8 |

```

Level 2:

Level completed! 🎉

Level 3 / 5 : add\_comm (level boss)

**Theorem** add\_comm: On the set of natural numbers, addition is commutative. In other words, if  $a$  and  $b$  are arbitrary natural numbers, then  $a + b = b + a$ .

```

theorem add_comm (a b : ℕ) : a + b = b + a := by
  1 induction b with d hd
  2 rw[add_zero, zero_add]
  3 rfl
  4 rw[add_succ, succ_add, hd]
  5 rfl
  6 |

```

Level 3:

Level completed! 🎉

Level 4 / 5 : add\_assoc (associativity of addition)

**Theorem** add\_assoc: On the set of natural numbers, addition is associative. In other words, if  $a, b$  and  $c$  are arbitrary natural numbers, we have  $(a + b) + c = a + (b + c)$ .

```

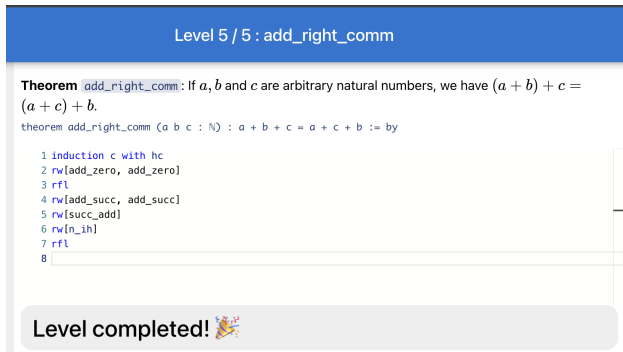
theorem add_assoc (a b c : ℕ) : a + b + c = a + (b + c) := by
  1 induction c with c hc
  2 rw[add_zero, add_zero]
  3 rfl
  4 rw[add_succ, add_succ]
  5 rw[add_succ]
  6 rw[hc]
  7 rfl
  8 |

```

Level 4:

Level completed! 🎉

Using induction on  $c$  we can initially create an easier medium to use reflexivity to solve for  $a+b$ . Then solving the other side we just use the mathematical definition of a successor function, until we can use the induction again to get the equation to the point where we can use reflexivity to prove it. This is a clear example of mathematical proofs by induction.



Level 5:

Once again this is proof by mathematical induction similar to the previous one. This time we add the zeroes and then use reflexivity for the first part of the proof. Then to prove the second part we use the successor function until bringing back the induction we used like the previous question. Then to complete the proof we use reflexivity again.

-

Discord Question: Since recursion has so many benefits and also breaks down the complexity of problems, why aren't we taught to use it as our primary method? In other words, why isn't it the first method of problem solving we're taught?

## 2.3 Week 3

### Notes

In week 3 we spoke about recursion further, and focused on our calculators in python. Eventually connecting the dots for our first Assignment that was due at the same time as Homework 4, where I used recursion in my python calculator in order to get it to function as efficiently as possible. We also discussed parsing, and derivation trees which is what we practiced in Homework 4.

### Homework

For homework 4 we did some practice on derivation trees for the strings you can see in the handwritten work below, along with the respective answers;

CPSC 354 - Programming Languages

Homework-4

Q) Write out the derivation trees for the following strings:

a)  $2+1$

A derivation tree for the expression 2+1. The root node is '+', which has two children: '2' and '1'.

b)  $1+2 \cdot 3$

Two derivation trees for the expression 1+2\*3. The left tree has root '+' with children '1' and '\*', where '\*' has children '2' and '3'. The right tree has root '\*' with children '+', '2', and '3', where '+' has children '1' and '2'.

c)  $1+(2 \cdot 3)$

A derivation tree for the expression 1+(2\*3). The root node is '+', which has children '1' and '\*'. The '\*' node has children '2' and '3'.

d)  $(1+2) \cdot 3$

A derivation tree for the expression (1+2)\*3. The root node is '\*', which has children '+', '2', and '3'. The '+' node has children '1' and '2'.

e)  $1+2 \cdot 3+4 \cdot 5+6$

A complex derivation tree for the expression 1+2\*3+4\*5+6. The root node is '+', which has children '1', '+', and '6'. The middle '+' node has children '2' and '\*'. The '\*' node has children '3' and '+'. This '+' node has children '4' and '\*'. The final '\*' node has children '5' and '3'.

### 3 Lessons from the Assignments

## References

7