CS 270 - Lab 7

M. Boady, B. Char, J. Johnson, G. Long

Week 7

1 Introduction

You may work in teams of up to two students. Submit one copy for the entire group. Answers must be submitted as a PDF. Answer all the questions in a word processor (like MS Word) and convert it to PDF. You may add *screenshots* of code or proofs as needed. You may not submit any handwritten answers. You do not need to copy the question text. Just label each question with it's question number/letter.

To insure you receive credit for your work in Gradescope, please insure that:

(a) all partners' names are included in your group submission

(b) your pages are tagged to the appropriate question it's answering

(c) your submission is legible (either typed or NEATLY written)

If we cannot read your answer to a question, it will not receive any points.
Graders will penalize submissions that do not follow the instructions.

Enter the name of the student in each role

Member 1: Nathan Xaysena

Member 2: Timmy Zheng

This lab is worth 100 points.

Question 1 : 6 points

Proofs by Induction are built on a simple concept. If we know a few predicates are true, and we know there is an implies for all other predicates, we can conclude something is true for an infinite set.

Imagine we have some predicate $S(x)$ that returns true or false for integer x. We also know the following.

$$S(1) \land S(2) \land \forall i \in \mathrm{N}\ (S(i) \Longrightarrow S(i+2))$$

With this statement, we can conclude that $\forall i \in \mathrm{N}\ (S(i))$. The statement must be true for all possible i.

Why is this enough to know $S(7)$ will be true? We know $S(1)$ is true. Since the for all is true for 1, we can create $S(1) \Longrightarrow S(3)$. We know that $S(3)$ is true because of $S(1)$. We can create another implies $S(3) \Longrightarrow S(5)$. We are now up to $S(5)$ being true. One more conditional reaches our goal. We create $S(5) \Longrightarrow S(7)$.

We now have a chain of implies.

$$S(1) \land (S(1) \Longrightarrow S(3) \Longrightarrow S(5) \Longrightarrow S(7))$$

We can also make a chain of implies for $S(8)$.

$$S(2) \land (S(2) \Longrightarrow S(4) \Longrightarrow S(6) \Longrightarrow S(8))$$

(a) (3 points) Use the following statement

$$S(1) \land S(2) \land \forall i \in \mathrm{N}\ (S(i) \Longrightarrow S(i+2))$$

Create a chain of implies that shows why $S(10)$ must be true.
S(2) ^ (S(2) -> S(4) -> S(6) -> S(8) -> S(10))

(b) (3 points) This time we will use a different starting statement.

$$M(1) \land \forall i \in \mathrm{N}\ (M(i) \Longrightarrow M(i+1))$$

Create a chain of implies that shows why $M(4)$ is true.

M(1) ^ (M(1) -> M(2) -> M(3) -> M(4))

Question 2 : 7 points

We will show the steps of a Proof by Induction by example.

We start with two Racket functions. One will be proven in this lab directions. Then you will use the same approach to prove the second one.

```
; Th is f u n c t i o n adds two numbers r e c u r s i v e l y
( d e f i n e ( add a b )
   ( i f (= b 0 ) a (+ 1 ( add a (− b 1 ) ) ) ) ) )
; Th is f u n c t i o n m u l t i p l i e s two numbers r e c u r s i v e l y
( d e f i n e ( mult a b )
   ( i f (= b 0 ) 0 (+ a ( mult a (− b 1 ) ) ) ) ) )
```

An inductive proof starts by evaluating the statement on a few fixed points. This gives us our $S(1) \wedge S(2) \wedge \cdots$. These are called the Base Case. We will only have one Base Case for this lab.

We want to prove that the recursive addition function works correctly. This function has two inputs. We start by fixing one and seeing what happens with Equational Reasoning.

```
; Proo f by I n d u c t i o n
; Argument : ( add a b ) = a+b f o r a l l i n t e g e r s a >= 0 , b >= 0
; For t h i s e n t i r e pr o o f , l e t x be a c o n s t a n t i n t e g e r x >= 0
; Base Case
;We p r ove t h a t ( add x 0) = x
; 1 . ( add x 0) ; Prem ise
; 2 . ( i f (= 0 0) x (+ 1 ( add x (− 0 1 ) ) ) ) ; Apply D e f i n i t i o n o f add ; 3 . ( i f #t x (+
1 ( add x (− 0 1 ) ) ) ) ; E v al u a t e Equ al s
; 4 . x ; E v al u a t e I f
; C onclu s i on t h e f u n c t i o n r e t u r n s x , wh ich i s t h e same as x+0s
```

(a) (7 points) You are going to prove that multiplication works. The Base Case is started for you. Complete the Equational Reasoning to finish the base case. You may type the answer or provide a screenshot from Racket.

```
; Proo f by I n d u c t i o n
; Argument : ( mult a b ) = a∗b for all integers a >= 0 , b >= 0
; For this entire proof , let x be a constant integer x >= 0
; Base Case
;We prove t h a t ( mul t x 0) = 0
```

1. **(mult x 0) premise**
2. **(if (= 0 0) 0 (+ x (mult x (- 0 1))))   Apply def of mult**
3. **(if #t 0 (+ x (mult x (- 0 1))))   Evaluate =**
4. **0   evaluate if**

Question 3 : 10 points

To create the implies ∀ *i* ∈ N (*M*(*i*) ==⇒ *M*(*i* + 1)) we need to make an assumption and see what happens. This is derived from the Conditional Introduction rule.

In a Proof by Induction, this is called the Inductive Case. The assumption we make is called the Inductive Hypothesis.

The Inductive Case and Inductive Hypothesis for the recursive addition is give below. Since we have already done the Base Case this concludes the entire proof.

; I n d u c t i v e Case
; I n d u c t i v e Hy p o t h e s i s :
; Assume f o r any i n t e g e r m in range 0 <= m <= k
; we know t h a t (add x m) = x+m

;We show t h i s IH c a u s e s (add x (+ k 1)) t o r e t u r n x+k+1
; 1 . (add x (+ k 1)) ; Prem ise
; 2 . (i f (= (+ k 1) 0) x (+ 1 (add x (− (+ k 1) 1)))) ; Apply D e f i n i t i o n o f add ; 3 . (i f #f x (+ 1 (add x (− (+ k 1) 1)))) ; E v al u a t e Equ al s
; 4 . (+ 1 (add x (− (+ k 1) 1))) ; E v al u a t e I f
; 5 . (+ 1 (add x k)) ; E v al u a t e S u b t r a c t i o n
; 6 . (+ 1 (+ x k)) ; By IH
; C onclu s i on : I f t h e IH i s t r u e t hen (add x (+ k 1)) w i l l r e t u r n x+k+1 ; By In d uc t i on , we have

show t h a t (add a b) = a+b f o r a l l i n t e g e r s a >= 0 , b >= 0

(a) (10 points) The inductive case for the multiplication function is started below. Finish it. You may type the answer or provide a screenshot from Racket.

; I n d u c t i v e Case
; I n d u c t i v e Hy p o t h e s i s :
; Assume f o r any i n t e g e r m in range 0 <= m <= k
; we know t h a t (mul t x m) = x∗m

;We show t h i s IH c a u s e s (mul t x (+ k 1)) t o r e t u r n (x ∗k+x)

; 1. (mult x (+ k 1)) Premise
; 2. (if (= (+ k 1) 0) 0 (+ x (mult x (- (+ k 1) 1)))) apply def of mult
; 3. (if #f 0 (+ x (mult x (- (+ k 1) 1)))) Evaluate =
; 4. (+ x(mult x (- (+ k 1) 1))) Evaluate If
; 5. (+ x (mult x k)) Evaluate -
; 6. (+ x (x k)) Apply IH*

Question 4 : 15 points

Prove by induction the following function returns a^b. Specifically, prove a base case x^0 and and inductive case (pow x k) = x^k ==⇒(pow x (+ k 1)) = $x * (x^k)$. You may use (exp x y) to mean the correct answer to x^yin Racket.

You may type the answer or provide a screenshot from Racket.

```
( d e f i n e ( pow a b )
    ( i f (= b 0 ) 1 (∗ a ( pow a (− b 1 ) ) ) )
)
```

```
; Proving the base case
; (pow x 0) ; premise
; ( if (= 0 0 ) 1 (* x ( pow x (- 0 1 ) ) ) ) ; apply def of pow
; (if #t 1 (* x (pow x (- 0 1)))) ; evaluate =
; 1 ; evaluate if

; Proving the inductive case
; Inductive hypothesis: Assume for 0 <= m <= k, (= (expr x k) (pow x k)))
; (pow x (+ k 1)) ; premise
; (if (= (+ k 1) 0 ) 1 (* x ( pow x (- (+ k 1) 1 ) ) ) ) ; apply def of pow
; (if #f 1 (* x (pow x (- (+ k 1) 1)))); evaluate =
; (* x (pow x (- (+ k 1) 1))); evaluate if
; (* x (pow x k)); evaluate -
; (* x (expr x k)) apply IH

; Now, prove (pow a b) => a^b
; (pow a b) ; Premise
; (* a (expr a (- b 1)) ; apply proof of induction for (pow x k)
; a * a^(b-1) Rewrite into algebraic expression
; a^b ; Evaluate *
```

Question 5 : 6 points

On paper, we can talk about numbers like 17 or 28. We learned this system of numbers and we all understand what it means.

If an alien were to come to earth and see the symbols 28 scratched on a wall, it would have no meaning to them. The symbol 28 also has no inherent meaning to a computer. On computer hardware, we traditionally use binary as our representation of numbers. The text value 28 is a representation of a number that is commonly used by humans.

When we want to prove things about how a computer uses numbers, we are really working with a representation. In a discrete math class, we could prove things about the concept of integers, but this would not reflect what happens on a computer.

When completing proofs related to programs, representations become crucially important. For example, an 8-bit computer can’t store 256 in memory. This changes how a program runs on an 8-bit vs 64-bit computer.

The first representation of numbers we will look at is called Peano Arithmetic. Peano Arithmetic is more colloquially called “pile of rocks” arithmetic. It is based on the idea that all arithmetic based on positive integers can be built from adding/removing rocks from a pile.

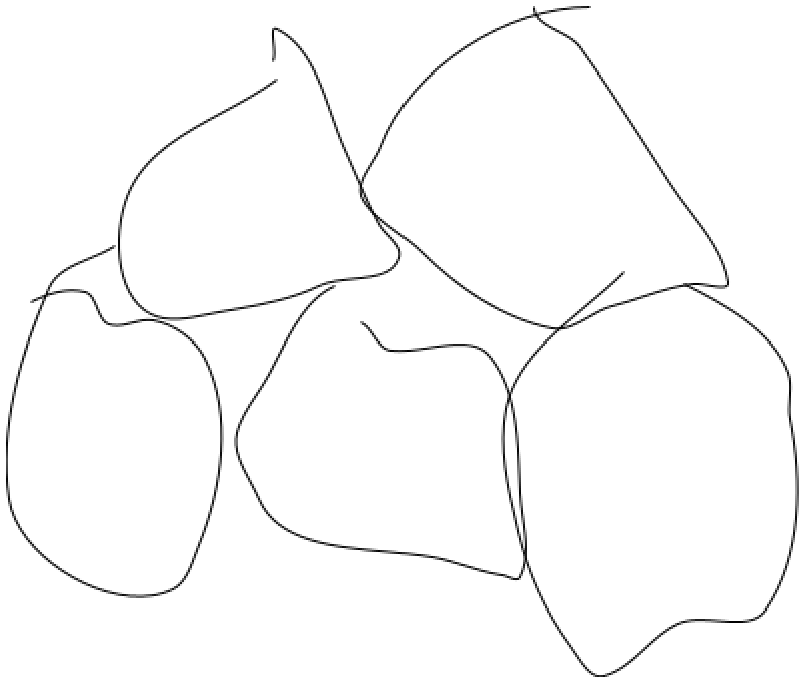
(a) (1 point) Are Roman Numerals a representation of numbers? Why or Why not?

Yes because it can be used to count something or used as an expression to represent the amount of something.

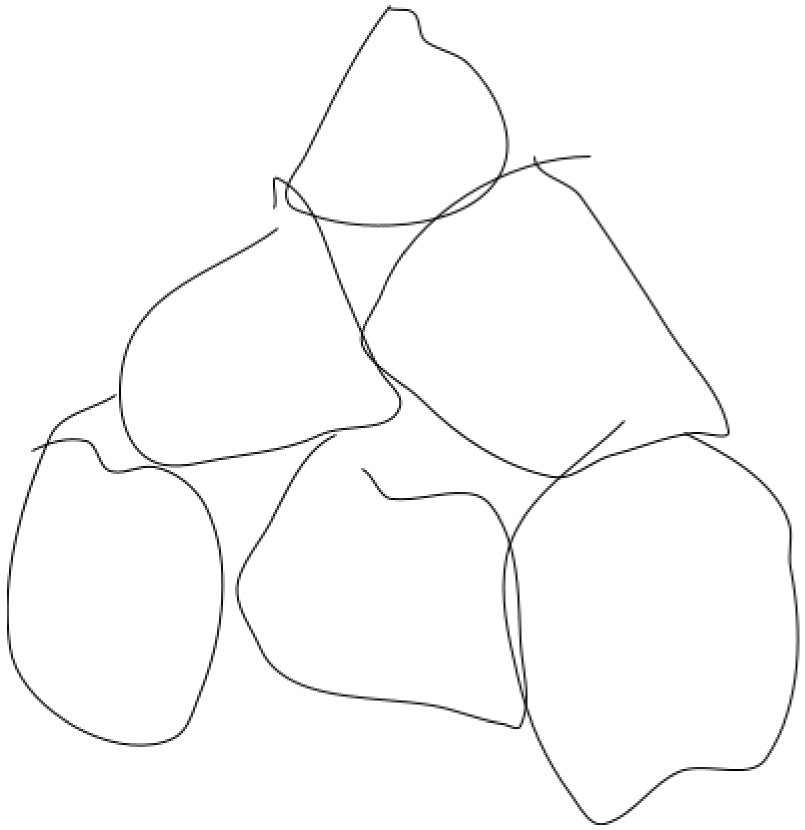
(b) (1 point) Write a number (of your choice) using three different representations.

4 (Hindu-Arabic numbers), IV (Roman numerals), llll (Tally marks)

(c) (1 point) Draw a pile of 5 rocks.



(d) (1 point) Draw one additional rock. (Just a single new rock)



(e) (1 point) How many total rocks have you draw on this page?

6

(f) (1 point) **I**s the answer to (e) the solution to 5 + 1?

[✓ Yes]
No

Question 6 : 10 points

Obviously, drawing rocks is a fun idea but doesn’t scale well. We will now use Racket Lists to build a more formal version of Peano Arithmetic.

The null list ‘() will represent zero, an empty pile with no rocks.

The s symbol ‘(s) will represent adding one rock to the pile p.

Representation	Rocks in (imaginary) Pile
‘()	0 rocks
‘(s)	1 rock
‘(s s)	2 rocks
‘(s s s)	3 rocks

All numbers can only be represented by this idea. Our symbol s represents a single rock. We create a list of s symbols to represent a number.

(a) (2 points) How many rocks are in the pile represented by ‘(s s s s s)?

5 rocks.

(b) (2 points) Give the Peano representation of a pile with 7 rocks.

‘(s s s s s s s)

(c) (3 points) How could we convert a Peano Number into a standard integer in racket?

(length '(s s s s s s s))

(d) (3 points) How could we convert a standard integer into a Peano Number?

Keep adding s into a list until there are a n amount of s inside the list to represent a positive integer n.

Question 7 : 5 points

We now have a basic structure to represent Peano Arithmetic in Racket.

We can build functions using our representation. First, we define a function to increment a number (add one to the number). We just need to cons an s onto the list.

```
; input−contract : a number in Peano Arithmetic representation
; output−contract : a number in Peano Arithmetic representation ( Answer to num + 1)
( define ( inc num) ( cons ’s num ) )
```

Prove: The Racket Expression (inc ’(s s)) returns ’(s s s). Use Equational Reasoning.   (inc '(s s))

(inc ‘(s s)) Premise

(cons 's '(s s)) Apply def of inc

'(s s s) Evaluate cons

Question 8 : 5 points

In our representation, negative numbers do not exist. The smallest number we can represent is 0. The following function decrements a number (subtracts one from the number).

```
; i n p u t−c o n t r a c t : a number in Peano A r i t hm e t i c r e p r e s e n t a t i o n
; o u t p u t−c o n t r a c t : a number in Peano A r i t hm e t i c r e p r e s e n t a t i o n ( Answer t o num − 1) (
d e f i n e ( dec num) ( i f ( nu l l ? num) nu l l ( rest num ) ) )
```

Prove: The Racket Expression (dec ’(s s)) returns ’(s).

(dec ‘(s s)) Premise

(if (null? ‘(s s)) null (rest ‘(s s))) Apply def of inc

(if #f null (rest ‘(s s))) evaluate null?

(rest ‘(s s)) evaluate if

‘(s) evaluate rest

Question 9 : 16 points

We can now begin building more complex mathematics. To do this, we need to think about both the representation and the arithmetic meaning. The representation determine how we write out program. The execution of the program still needs to follow the arithmetic meaning.

Right now, we have the following abilities in our Peano Arithmetic

1. Represent Positive Integers
2. Add 1 to a number (increment)
3. Subtract 1 from a number (decrement)

Using just these commands, how could we compute 3 + 2?

(a) (2 points) Imagine we have two piles of rocks. Pile A has 3 rocks and Pile B has 2 rocks.

If we take one rock out of pile B and place it in pile A, how many rocks are in each pile?

Pile A has 4 rocks. Pile B has 1 rock.

(b) (2 points) Continuing from your answer to the previous question.

If we take another rock out of pile B and place it in pile A, how many rocks are in each pile?

Pile A has 5 rocks. Pile B has 0 rocks.

(c) (2 points) At this point, one pile of rocks should contain the answer to 3 + 2. Which pile?

Pile A contains the answer to 3 rocks + 2 rocks or 5 rocks.

(d) (2 points) What is the result of computing (4 + 8)?

12

(e) (2 points) What is the result of computing (1 + (4 + (8 − 1)))?

12

(f) (2 points) What should the base case of s recursive addition function be?

The base case should address when the Peano value of the list being added to another list has no more symbols to decrement from, or when there are no more rocks in the pile that's being added to another.

(g) (4 points) Describe in plain English how the recursive case of addition should work? You may use algebraic expressions in your answer.

A recursive case of addition would be repeated decrements of by 1 for the number being appended to the other number, which will receive repeated increments of 1. To perform this repeated decrement, one can start by doing b-1, then taking that result and performing the same operation on it recursively with ((b-1)-1) and (((b-1) -1) -1) until b decrements to 0. Meanwhile, the repeated addition would also be recursive in a similar fashion as decrementing, where the operation of a+1 will be applied recursively for the values of ((a+1)+1) and (((a+1)+1)+1) until b becomes 0.

Question 10 : 20 points

We can now define addition using the method we worked out on the previous page.

```
; input−contract : two numbers in Peano Arithmetic representation
; output−contract : a single number in Peano Arithmetic representation (Answer to A+B)
(define (add A B)
  (if (null? B)
      A
      (inc (add A (dec B)))))
```

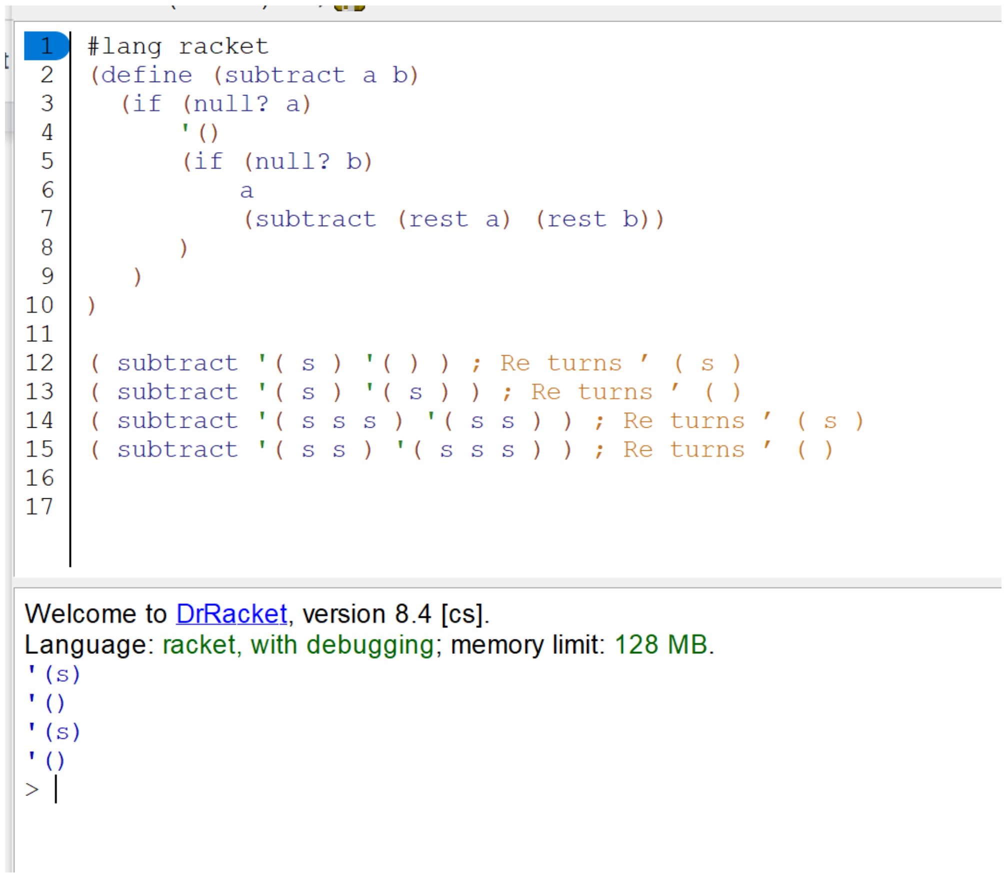
This implementation is based on the reasoning from the previous page.

Write a recursive function for subtraction (subtract A B) that computes $A - B$ in Peano Arithmetic. We have no negative numbers, return 0 if the solution is negative.

Here are some example inputs/outputs to test with.

Use the solution and reasoning behind add to come up with your code for subtract. You may not convert the list to an integer. Use the structure of the list. Attach a screenshot of your code.

```
(subtract '(s) '()) ; Returns '(s)
(subtract '(s) '(s)) ; Returns '()
(subtract '(s s s) '(s s)) ; Returns '(s)
(subtract '(s s) '(s s s)) ; Returns '()
```



```
#lang racket
(define (subtract a b)
  (if (null? a)
      '()
      (if (null? b)
          a
          (subtract (rest a) (rest b))
      )
  )
)

( subtract '( s ) '( ) ) ; Re turns ' ( s )
( subtract '( s ) '( s ) ) ; Re turns ' ( )
( subtract '( s s s ) '( s s ) ) ; Re turns ' ( s )
( subtract '( s s ) '( s s s ) ) ; Re turns ' ( )
```

```
Welcome to DrRacket, version 8.4 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'(s)
'()
'(s)
'()
> 
```