CS270: LAB #5

pNums (Peano Numbers)

You may work in teams of 1-2 people (three is acceptable in the event of an unscheduled absence). Unless stated otherwise, the lab is due to be submitted into Gradescope at the end of the day if not done by the end of class. In order to receive credit, follow these instructions:

[a] Every team member should be discussing simultaneously the same problem – do NOT try to divvy up the labor and assign different problems to different students since the material is cumulative.

[b] Directly edit this lab PDF using Sedja/PDFescape with your answers (extra pages can be added in the rare event you need more than the allotted space)

[c] Each lab, rotate which member has the responsibility of being the Scribe. This is the person that is typing the answers and uploading the final PDF – note that only a single copy of the filled in PDF is turned into Gradescope. Only one lab needs to be submitted for the entire team, and all members receive the same score. Make sure to use a font that your PDF editor is compatible with (otherwise you might find your answers appear as weird shapes/sizes or simply disappear entirely!)

[d] The Gradescope submission must have each answer properly tagged with the appropriate question. Moreover, every member of the team must be listed as a submitter. Although it is the Scribe which executes these actions, it is still the responsibility of the entire team to make certain this is done properly (thus it is highly recommended that the Scribe share their screen so the entire team can witness it). Answers which are improperly tagged cannot be seen by the grader and thus cannot be scored.

[e] **FOR REMOTE ONLY**: Each lab, rotate which member has the responsibility of being the Recorder. This is the person who hits the Zoom Record button (once the technical permission is granted by the TA/RCF/Professor) and ensures that everyone has their camera/microphone on. They are also the member that is responsible to make sure the DrexelStream video is marked as viewable and entered into the https://tinyurl.com/VidLinkForm webform before 11:59pm (they should also email the rest of their team as confirmation.) Note that the video file doesn't get created/processed until after the Recorder has quit Zoom.

[f] Each lab, rotate which member has the responsibility of being the Manager. This is the person that ensures that everyone is participating equally and honestly, keeps the group on task, ensures that all team members understand a solution before going on to the next question, and presses the "hand up" button in Zoom to summon a TA or the professor (but they only do so after surveying the group to make sure everyone has the same question).

Team Name (CS pioneer): Tim Berners-Lee

Scribe name: Jeremy Mathews

Recorder name: Cole Bardin

Manager name: Jackson Masterson

Other team member (if any): Brendan Hoag

Question 1 : 15 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.

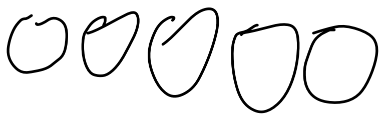
(a) (3 points) Are Roman Numerals a **representation** of numbers? Why or Why not?

Yes, because they are used to make calculations, create bigger numbers, and are designed to be easily readable and writable by human beings.

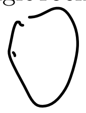
(b) (3 points) Write a number (of your choice) using three different representations.

50, L, 110010

(c) (3 points) Draw a pile of 5 rocks.



(d) (2 points) Now draw a single rock.



(e) (2 points) How many total rocks have you drawn on this page?

6

(f) (2 points) Is the answer to (e) the solution to $5 + 1$?

[✓ Yes]
No

Question 2 : 9 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 constant z will represent zero, an empty pile with no rocks.

The Racket list (s p) will represent adding one rock to the pile p.

representation	Rocks in (imaginary) Pile
z	0 rocks
(s z)	1 rock
(s (s z))	2 rocks
(s (s (s z)))	3 rocks

All numbers can only be represented by this idea. Our function s is a function with 1 input. That means s means nothing by itself. Also, a command like (s s s (s z)) means nothing in our representation.

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

5 rocks

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

we call this abstraction of the integers, "pnums"

(s (s (s (s (s (s (s z)))))))

(c) (1 point) What does the command (**list** 's '(s (s z))) do?
- ○ Nothing in Peano Arithmetic
- [✓ Compute 2 + 1 = 3 in Peano Arithmetic]
- ○ Compute 2 − 1 = 1 in Peano Arithmetic

(d) (1 point) What does the command (**cons** 's '(s (s z))) do?
- [✓ Nothing in Peano Arithmetic]
- ○ Compute 2 + 1 = 3 in Peano Arithmetic
- ○ Compute 2 − 1 = 1 in Peano Arithmetic

(e) (1 point) What does the command (**first** '(s (s z))) do?
- [✓ Nothing in Peano Arithmetic]
- ○ Compute 2 + 1 = 3 in Peano Arithmetic
- ○ Compute 2 − 1 = 1 in Peano Arithmetic

(f) (1 point) What does the command (**second** '(s (s z))) do?
- ○ Nothing in Peano Arithmetic
- ○ Compute 2 + 1 = 3 in Peano Arithmetic
- [✓ Compute 2 − 1 = 1 in Peano Arithmetic]

(g) (1 point) What does the command (**rest** '(s (s z))) do?
- ○ Nothing in Peano Arithmetic
- ○ Compute 2 + 1 = 3 in Peano Arithmetic
- [✓ Compute 2 − 1 = 1 in Peano Arithmetic]

Question 3 : 10 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 (i.e. increases a number by one)

; input–contract: N *is the Peano Arithmetic representation of the non-negative integer n*
; output–contract: (inc N*) is the pnum representation of the integer n+1*

(define (inc N) (list 's N))

Use **equational reasoning to prove that** Racket Expression **(inc** '(s (s z))) **returns** '(s (s (s z))).

(inc '(s (s z)))	Premise
(list 's '(s (s z)))	Definition of inc
'(s (s (s z)))	Evaluate list

Question 4 : 10 points

In our representation, negative numbers do not exist. The smallest number we can represent is 0.

The following function decrements a number (i.e. decreases the number by one).

; input–contract: N *is a Peano representation of the number n, and n is not* 0.
; output–contract : (dec N*) is the Peano representation of n-1*

(define dec second)

[a] What would happen if we tried to decrement zero? In other words, if we tried to evaluate (dec 'z)

Why do we not actually have to concern ourselves with that situation?

Running dec on `z just causes an error saying that there are too few arguments.

[b] use equational reasoning to prove that decrementing (the representation of) 2 gives you (the representation of) 1

```
(dec `(s (s z))) ; premise
(second `(s (s z))) ; Apply Def. of dec
`(s z) ; Evaluate second
```

Question 5 : 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 and Pile B has 1 rock

(b) (2 points) Continuing from your answer to the previous question. 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 5 rocks and 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+2

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

(4 + 8) = 12

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

(4 + 1) + (8 - 1) = 5 + 7 = 12

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

The base case should be when one pile has no rocks, return the other pile

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

The recursive case for addition should be calling the addition function add(X Y) on (X+1) and (Y-1), just as we did in part e.

Question 6 : 20 points

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

```
; input-contract: A,B are Peano numbers reprsenting the integers a,b.
; output-contract: (add A B) is the Peano number representing a+b.
(define (add A B)
  (if (equal? B 'z)
      A
      (add (inc A) (dec B))))
```

This implementation is based on the reasoning from the previous page. We will now verify that our function works as expected. This time, we need to apply all our function definitions. We are verifying the code actually works.

Prove: The Racket Expression (add '(s z) '(s z)) returns '(s (s z)). You **must** apply all three user defined functions (inc, dec, and add).

(add '(s z) '(s z)) Premise

(if (equal? '(s z) 'z) '(s z) (add (inc '(s z)) (dec '(s z))))) Apply definition of add

(if (#f) '(s z) (add (inc '(s z)) (dec '(s z))))) Evaluate equal?

(add (inc '(s z)) (dec '(s z)))) Evaluate if statement

(add (list 's '(s z)) (dec '(s z)))) Apply definition of inc

(add '(s (s z)) (dec '(s z))) Evaluate list function

(add '(s (s z)) (second '(s z))) Apply definition of dec

(add '(s (s z)) 'z) Evaluate second

(if (equal? 'z 'z) '(s (s z)) (add (inc '(s (s z))) (dec 'z))) Apply definition of add

(if (#t) '(s (s z)) (add (inc '(s (s z))) (dec 'z))) Evaluate equal?

'(s (s z)) Evaluate if statement

Question 7 : 20 points

Write a recursive function that represents subtraction of Peano representations, A,B.
Since we don't yet have a representation of negative numbers, return 'z for (subtract A B) if a < b

Here are some example inputs/outputs to test with.

Your function definition **must** include comments with the input and output contracts.

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

```
; input contract: a and b are peano numbers representing integers a, b

; output contract: (subtract a b) is the peano number representing a-b

(define (subtract a b)
 (cond
  [(equal? a 'z) 'z]
  [(equal? b 'z) a]
  [else (subtract (dec a) (dec b))]
  ))
```