CS270: LAB #17

Code Correctness Proofs

You may work in teams of ideally one or two 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 class day (11:59pm).
to allow for upload time etc). 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):	Marvin Minsky
Scribe name:	Jerry Li
Recorder name:	Evelyn Thai
Manager name:	Terie Ha
Other team member (if any):	Lixiao Yang

Question 1 : 20 points

Completing **Induction Proofs** on data structures is more complicated. We won't have algebraic knowledge to fall back on. We can still complete proofs using programming logic.

These proofs will require more detailed assumptions to me made. Before starting on proofs, we will review some concepts about lists.

Let L be some Racket list. We don't know anything else about L (it might even be the empty list).

Let a be some Racket object. We don't know anything else about a (it might be an integer, it might be list).

(a) (4 points) What will (**null**? (**cons** a L)) return? Why?

It should return false since the list is not empty.

(b) (4 points) What will (**first** (**cons** a L)) return? Why?

It should return a because a is the first element in the list.

(c) (4 points) What will (**rest** (**cons** a L)) return? Why?

It should return '(L) because '(L) is the rest after we drop first element in the list.

(d) (4 points) Let us further assume (**length** L) is equal to some integer $x \geq 0$.
What is (**length** (**cons** a L))? Does the value of a matter to the answer?

It should return x+1 because the length of L is x. The value of a doesn't matter since it's just an element.

(e) (4 points) What is (**length** (**rest** L)) in terms of x? Are there any possible L that could cause an error here?

It should return x-1 because the length of the L is x. We drop the first element so length of rest L should be x-1.
When L is an empty list, it would cause an error here since you cannot get rest of null.

Question 2 : 16 points

Review the following function.

```
; input−contract: L is a list of integers
; output−contract: (squareAll L) is the same as the original list but with all elements squared
; Example: (squareAll '(3 1 2)) would be '(9 1 4)

(define (squareAll L)
    (if(null? L) null(cons(*(firstL) (firstL))(squareAll(rest L)))))
```

(a) (6 points) The smallest possible list is the null list. The null case will act as our base case. Complete the following LHS Equational Reasoning proof.

1. (squareAll null)	Premise of LHS (with L as null)
2. (if(null? null) null (cons(*(first null)(first null))(squareAll(rest null)))))	apply def. of squareAllnull
3. (if #t null (cons(*(first null)(first null))(squareAll(rest null)))))	evaluate null?
4. null	evaluate if

(b) (4 points) You should have ended with null in part (a). Explain why this matches the RHS.

RHS is the square of all the element in the list. Since the list is null, the square of the list should be null.

(c) (6 points) Explain why any list of integers can be created using only the null list, integers, and the cons command.

We can start with null and cons any integers to the list.

Question 3 : 12 points

In the previous question, you provided a **Base Case** for the **Inductive Proof**.

The Inductive Hypothesis will also be more complicated.

Inductive Hypothesis

Suppose we have a list of integers L and call (squareAll L) = M. Our IH is that the function works properly on L, In other words, the IH is that M is a list of the squares of L (in the same order). Our goal for the leap step is to prove that the function works on the "next largest" list. in other words, that (squareAll (cons a L)) is a list of all the squares of (cons a L)

(a) (4 points) As far as we know right now, there is only 1 list for which the function works. What is the one list L that we currently for sure know that the function works for?

The base case, the null list.

(b) (4 points) Explain in your own words how (cons a L) is the "next largest" list after L, where a is some integer? **In what ways is this step of structural induction similar to induction on the integers from the last lab? In what ways is it different?**

Because the length of (cons a L) is x+1 similar to last lab k+1.

For last lab, we were doing operations on intergers and this lab we are doing operations on length of the list.

(c) (4 points) We don't know what values are in L, so we can't predict what the result of (squareAll L). We can assume it worked, but that doesn't mean we know the exact list that would be returned by the function. We refer to the output by (squareAll L) by the name M.

What will (**equal**? (**length** L) (**length** M)) return? Why?

True, because the length of the list remain the same after square. The value in the list doesn't matter.

Question 4 : 12 points

Review the following function.

```
; input− contract : L is a list of integers
; output− contract : (squareAll L) is the same as the original list but with all elements squared
; Example: (squareAll '(3 1 2)) would be '(9 1 4)

(define (squareAll L)
    (if (null? L) null (cons (* (first L) (first L)) (squareAll (rest L)))))
```

IH: (squareAll L) = M, where M is the list of squares of the members of L, in the same order

LEAP Step: Let a be any integer.

LHS:

1.	(squareAll (cons a L))	Premise of LHS
2.	(if (null? (cons a L)) null (cons (* (first (cons a L)) (first (cons a L))) (squareAll (rest (cons a L)))))	Apply definition of squareAll
3.	(if #f null (cons (* (first (cons a L)) (first (cons a L))) (squareAll (rest (cons a L)))))	Eval null?
4.	(cons (* (first (cons a L)) (first (cons a L))) (squareAll (rest (cons a L))))	Eval If
5.	(cons (* a a) (squareAll (rest (cons a L))))	Eval first-cons (twice)
6.	(cons (* a a) (squareAll L))	Eval rest-cons
7.	(cons (* a a) M)	By IH

(a) (4 points) Why did null evaluate to false on line 3?

Because we cons a integer to a list L, the list L would never be empty. So the evaluation of (null? (cons a L)) will be false.

(b) (4 points) What does it mean that first and cons cancel out on line 5?

When you cons 'a' into the list and add it to the begining, then taking first element would be 'a'.

(c) (4 points) What would the RHS for this proof be? Explain why it matches with what the LHS got on line 7.

Result of RHS should be (cons a*a L*L). The first element has matched the LHS, and L*L is equal to M since M = (squareAll L)

Question 5 : 20 points

Review the following function.

```
; input-contract: L is any list
; output-contract: (count L) is a nonnegative integer which is the number of members in L
; Example: (count '(a b c)) would be 3

(define (count L)
    (if (null? L) 0 (+ 1 (count (rest L)))))
```

(a)(6 points) Base Case: Prove that LHS of (count null) returns the same as the RHS, which is the num of items in null.

1. (count null)	premise of LHS
2. (if (null? null) 0 (+ 1 (count (rest null)))))	apply def. of count
3. (if #t 0 (+ 1 (count (rest null)))))	evaluate null?
4. 0	evaluate if

RHS: Since the list is empty, there is 0 nonnegative integer in L. The result of RHS would be 0.

(b) (4 points) Complete the Inductive Hypothesis by filling in the rest of the sentence after the "..."

(count L) = k, where k is .... the number of members in L

(c) (10 points) Complete the Leap Step

Let a be a valid racket object.

LHS should start with (count(cons a L)) as the premise, RHS would be the number of items in (cons a L)

You must write an equational reasoning proof which demonstrates that the LHS = RHS

LHS

1. (count (cons a L))	premise of LHS
2. (if (null? (cons a L)) 0 (+ 1 (count (rest (cons a L))))))	apply def. of count
3. (if #f 0 (+ 1 (count (rest (cons a L))))))	evaluate null?
4. (+ 1 (count (rest (cons a L)))))	evaluate if
5. (+ 1 (count L))	evaluate rest-cons
6. k+1	by IH

RHS

(k + 1)	premise of RHS: replace k by k+1
k+1	algebra

Since LHS=RHS (both are k+1), the Leap is established.
Thus both the base case and the leap have been demonstrated,
And consequently it has now been proved by induction that
(count (L)) is the number of members in L for all lists

Question 6 : 20 points

Review the following function.

```
; input-contract: L is a list of positive integers
; output-contract: (doubleAll L) is the same list as L but with each value doubled.
; Example: (doubleAll '(3 1 2)) would be '(6 2 4)
(define(doubleAll L)
   (if (null? L) null (cons (* 2 (first L)) (doubleAll (rest L)))))
```

Prove: That this function doubles every value in a list of integers.

Scoring: 5pts Base Case with conclusion, 5pts Inductive Hypothesis statement, 10pts Leap Step and conclusion

Base case: the list is empty

LHS: (doubleAll null)

1. (doubleAll null)	premise of LHS
2. (if (null? null) null (cons (* 2 (first null)) (doubleAll (rest null)))))	apply def. of doubleAll
3. (if #t null (cons (* 2 (first null)) (doubleAll (rest null)))))	evaluate null?
4. null	evaluate if

RHS: note that since L = null, (doubleAll L) = null

Since LHS=RHS, the base case has been established

IH: (doubleAll L) = M, where M is the list of double of the members of L

1. (doubleAll (cons a L))	premise of LHS
2. (if (null? (cons a L)) null (cons (* 2 (first (cons a L))) (doubleAll (rest (cons a L))))))	apply def. of doubleAll
3. (if #f null (cons (* 2 (first (cons a L))) (doubleAll (rest (cons a L))))))	evaluate null?
4. (cons (* 2 (first (cons a L))) (doubleAll (rest (cons a L)))))	evaluate if
5. (cons (* 2 a) (doubleAll (rest (cons a L)))))	evaluate first-cons
6. (cons (* 2 a) (doubleAll L))))	evaluate rest-cons
7. (cons (* 2 a) M)	by IH

RHS:

Because (doubleAll (cons a L)) = (cons (doubleAll a) (doubleAll L)),
we replace M with (doubleAll L) and (doubleAll a) = (* 2 a) so (cons (doubleAll a) (doubleAll L)) == (cons (* 2 a) M))