CS 270 - Lab 13

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

1 Introduction

You may work in teams of **one** or **two** students. Submit one copy for the entire group.

Write your answers on this lab sheet. Only what is written on this lab sheet will be graded.

This lab is due at the end of the class period. You may not continue to work on it once class has ended.

This lab contains 4 questions.

Grading

- 25 points - Putting everyones names on this page
- 20 points - Earned for each correct question (Answer is fully correct)
- 5 points - Earned for **partial credit** any question

No additional point amounts can be earned. You cannot earn 7 points on a question for example.

The maximum score for a lab is 100. If you get everything correct, that adds up to 105 points but will be reduced to 100.

A question will be marked correct as long as it covers all requirements of the question. It does not need to be perfect, but must be fully correct. A single typo or very minor issue where the intention is clear and all requirements are met would still earn full points.

We want you to complete questions fully, not try to earn partial credit on multiple questions. You may ask your Professor/Course assistant questions during lab.

Labs must be done in the presence of an instructor and/or course assistant or credit will not be given.

Partners should alternate each class day which person is physically typing and submitting the lab.

Do not split up the problems or you risk not finishing on time due to the cumulative nature of the questions.

Enter the name of the student in the group

Member 1 (submitter): Elan Rubin

Member 2: Joshua Koo

Question 1 :

Jeb has a set of lists that are his favorite.s In this question, you will try to figure out what if lists are part of the set.

The set of lists in Jeb's favorites are defined recursively.

You know the following:

- The null list is in the set of Jeb's **favorites**
- If you take any list in the set of **favorites** and you add a vowel to the list, it will still be in the set of **favorites**.

Decide if each of the following is one of Jeb's favorites.

(a) Is list '(s a b b a c c) in the set of Jeb's favorites?

- ○ Yes
- ⊗ No

(b) Is list '(a e i o u) in the set of Jeb's favorites?

- ⊗ Yes
- ○ No

(c) Is list '(a a a a a a) in the set of Jeb's favorites?

- ⊗ Yes
- ○ No

(d) Is list '(e e e s e) in the set of Jeb's favorites?

- ○ Yes
- ⊗ No

(e) Make a list with 7 elements that you know will be one of Jeb's favorites.

'(a e i a a a a)

there are many, many different answers that are possible

(f) How many lists meet the requirements of being one of Jeb's favorites?

there is an infinite number of lists that meet the requirements

Question 2 :

Imagine we want to do an Inductive Proof on the following function.

```
;input-contract: A list M
;output-contract: the list M in reverse order
(define (reverse M)
  (if (null? M)
      M
      (append (rest M) (list (first M)))))
```

Let L be the set of all lists in Racket.

Claim: $\forall x \in L$ ((length (reverse x)) = (length x))

(a) What is the anchor case? (you do not need to prove it)

the anchor case is a null list, '()

(b) Base Case LHS Premise (you do not need to prove it)

the base case LHS premise would be (length (reverse null))

(c) Base Case RHS Premise? (you do not need to prove it)

(length null)

(d) What is the Inductive Hypothesis? (you do not need to prove it)

; Inductive Hypothesis
; Let us assume there is list L
; (length L) = n
; then we have (length (reverse L)) = (length L)

(e) What is the Leap Case LHS Premise? (you do not need to prove it)

(length (reverse (cons k L)))

(f) What is the Leap Case RHS Premise? (you do not need to prove it)

(length (cons k L))

Question 3 :

```
;input−contract: An integer a >= 0
;output−contact; A list containing a copies of integer 2
(define (seq a) (if (< a 1) null (cons 2 (seq (− a 1)))))
```

Claim: $\forall x \in \mathbb{Z}\,(x >= 0 \implies$ (length (seq x)) = x)

You may use the following Lemmas.

Lemma 1: (**length null**) = 0, **Lemma 2**: (**length** (**cons** a L)) = (+ 1 (**length** L))

```
;Base Case (anchor at n=0)
;LHS
(length (seq 0));Premise of LHS
(length (if (< 0 1) null (cons 2 (seq (− 0 1)))));Apply seq
(length (if #t null (cons 2 (seq (− 0 1)))));Eval =
(length null);eval if
0;by Lemma 1
;RHS
0;Premise of RHS
;Since LHS = RHS we have established the base case.
```

State the IH and then prove the Leap Case

IH: (length (seq k) = k)

LHS:
(length (seq (+ 1 k)) ; premise of LHS leap case
(length (if (< (+ k 1) 1) null (cons 2 (seq(- (+ k 1) 1))) ;apply definition of seq
(length (if (#f) null (cons 2 (seq(- (+ k 1) 1))) ;evaluation of <
(length ((cons 2 (seq(- (+ k 1) 1)) ;evaluation of if
(length ((cons 2 (seq(k)) ;evaluation of - and +
(+ 1 (length (seq k))) ; lemma 2
1 + k ;invoke the IH
k+1 ;algebra

RHS:
k +1 ;premise of RHS leap case

```
;The Leap Case and Base Case are established, thus by POMI
;For All x in Z (x >= 0 −> (length (seq k)) = k )
```

Question 4 :

```
;input-contract: L is a list containing only integers or null list
;output-contract: (total L) is an integer (the sum of all values in L)
(define (total L) (if (null? L) 0 (+ (first L) (total (rest L)))))
```

Let L be the set of all lists containing only integers and the null list.

Claim $\forall x \in \mathbb{L}$ ((integer? (total x)) = #t)

Prove this claim using induction. You may use the following Lemma.

Lemma 1: (integer? (+ a b)) = (**and** (integer? a) (integer? b))

You may write on the back of this page if needed.

Anchor at L = '()

Base Case LHS

(integer? (total null)) ; Premise Of Base case LHS
(integer? (if (null? '()) 0 (+ (first null) (total (rest null)))))); apply def of total
(integer? (if #t 0 (+ (first null) (total (rest null))))) ; Evaluate of null?
(integer? 0) ; Evaluate if
#t ; Evaluate integer?

Base Case RHS

#t ; Premise of base case RHS

#t = #t

IH = (integer? (total k)) = #t

Leap Case:
LHS:
(integer? (total (cons k L)); Premise of leap case LHS
(integer? (if (null? (cons k L)) 0 (+ (first (cons k L) (total (rest (cons k L)))) ; apply def of total
(integer? (if #f (+ (first (cons k L) (total (rest (cons k L))))) ; evaluate null?
(integer? (+ (first (cons k L) (total (rest (cons k L))))) ; evaluate if
(integer? (+ k (total (rest (cons k L))))) ; Evaluate first cons
(integer? (+ k (total L))) ; Evaluate rest and cons
(and (integer? k) (integer? (total L)); Lemma 1
(and #t (integer? (total L))) ;Evaluate integer?
(and #t #t) ; Invoke IH
#t ; evaluate and

RHS:
#t ; premise of RHS