CS 61A Structure and Interpretation of Computer Programs
Spring 2016

Discussion Quiz 9

1. (3 points) Chasing Tails

Identify whether or not each of the following procedures uses a constant amount of space in a tail-recursive Scheme implementation (i.e. whether **every** recursive call is a tail call).

```
(define (copy lst result)
    (if (null? lst) result
        ((lambda (copy) copy) (copy (cdr lst)
                                    (append result (list (car lst)))))))
```

(Note: `append` takes two or more lists and constructs a new list with all of their elements.)

```
(define (broken lst) (broken (broken lst)))
```

```
(define (is-ascending lst last-num)
    (if (null? lst) #t
        (and (is-ascending (cdr lst) (car lst)) (> (car lst) last-num))))
```

(This subroutine would need to be called with a `last-num` that is less than all of the elements in the list.)

---

2. (3 points) It's Hailing... Again

Write a *tail-recursive* version of `hailstone`. This procedure accepts a positive integer `n` and an empty list `lst`, and returns a list containing the hailstone sequence that starts at `n`.

As an example, `(hailstone 5 '())` would return `(5 16 8 4 2 1)`.

```
(define (hailstone n lst)

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------)
```

3. (4 points) Humans Need Not Apply

What does `eval` do (in the context of an interpreter)? What does `apply` do?