CS 61A Structure and Interpretation of Computer Programs

Spring 2017 DISCUSSION QUIZ 9

1. (3 points) Heads or 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)))))))
```

(Remember that `append` takes zero or more lists and constructs a new list with all of the lists' 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))))
```

(Assume that this procedure is always called with a `last-num` that is less than all of the elements in the list.)

```
-------------------------------------------------------------------------
```

2. (4 points) Hail Recursion

Write a *tail-recursive* version of `hailstone`. This procedure accepts a positive integer `n` and returns a list that contains the hailstone sequence starting at `n`. For instance, `(hailstone 5)` would return `(5 16 8 4 2 1)`.

```
(define (hailstone n)
  (define (hs-helper n lst)

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

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

  ----------------------------------------------------------------------)

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

3. (3 points) Humans Need Not Apply

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