TAIL RECURSION AND STREAMS

COMPUTER SCIENCE MENTORS 61A

April 10 to April 14, 2017

1 Tail Recursion

1. What is a tail context/tail call? What is a tail recursive function?

2. Why are tail calls useful for recursive functions?

Answer the following questions with respect to the following function:

```
(define (sum-list lst)
  (if (null? lst)
    0
    (+ (car lst) (sum-list (cdr lst)))
  )
)
```

3. Why is sum-list not a tail call? Optional: draw out the environment diagram of this sum-list with list: (1 2 3). When do you add 2 and 3?

4. Rewrite sum-list in a tail recursive context.

2 Streams

5. Whats the advantage of using a stream over a linked list?

6. Whats the maximum size of a stream?

7. Whats stored in first and rest? What are their types?

8. When is the next element actually calculated?

3 What Would Scheme Print?

9. For each of the following lines of code, write what scheme would output.

```
scm> (define x 1)


scm> (if 2 3 4)


scm> (delay (+ x 1))


scm> (define (foo x) (+ x 10))


scm> (define bar (cons-stream (foo 1) (cons-stream (foo 2)
   bar)))
```

```
scm> (car bar)



scm> (cdr bar)



scm> (define (foo x) (+ x 1))


scm> (cdr-stream bar)


scm> (define (foo x) (+ x 5))


scm> (car bar)


scm> (cdr-stream bar)
```

4 Code Writing for Streams

10. Write out double_naturals, which is a stream that evaluates to the sequence 1, 1, 2, 2, 3, 3, etc.

```
(define (double_naturals)
    (double_naturals_helper 1 0)
)

(define (double_naturals_helper first flag)















)
```

11. Write out interleave, which returns a stream that alternates between the values in stream1 and stream2. Assume that the streams are infinitely long.

```
(define (interleave stream1 stream2)















)
```