CS 61A Summer 2009 Week 5B Lab
Wednesday 7/22 Afternoon

1. What is the type of the value of `(delay (+ 1 27))`? What is the type of the value of `(force (delay (+ 1 27)))`?

2. Evaluation of the expression

```
(stream-cdr (stream-cdr (cons-stream 1 '(2 3))))
```

produces an error. Why?

3. Consider the following two procedures.

```
(define (enumerate-interval low high)
  (if (> low high)
    '()
    (cons low (enumerate-interval (+ low 1) high)) ) )

(define (stream-enumerate-interval low high)
  (if (> low high)
    the-empty-stream
    (cons-stream low (stream-enumerate-interval (+ low 1) high)) ) )
```

What's the difference between the following two expressions?

```
(delay (enumerate-interval 1 3))
(stream-enumerate-interval 1 3)
```

4. An unsolved problem in number theory concerns the following algorithm for creating a sequence of positive integers $s_1, s_2, ...$

> Choose s_1 to be some positive integer.
> For $n > 1$,
> if s_n is odd, then s_{n+1} is $3s_n + 1$;
> if s_n is even, then s_{n+1} is $s_n/2$.

No matter what starting value is chosen, the sequence always seems to end with the values 1, 4, 2, 1, 4, 2, 1, ... However, it is not known if this is always the case.

4a. Write a procedure `num-seq` that, given a positive integer `n` as argument, returns the stream of values produced for `n` by the algorithm just given. For example, `(num-seq 7)` should return the stream representing the sequence 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, ...

4b. Write a procedure `seq-length` that, given a stream produced by `num-seq`, returns the number of values that occur in the sequence up to and including the first 1. For example, `(seq-length (num-seq 7))` should return 17. You should assume that there is a 1 somewhere in the sequence.

NOTE: The rest of this lab will deal with the metacircular Evaluator. All of the exercises below will be on homework 5 also.

5. Copy the file ~`cs61a/lib/mceval-step1.scm` to your home directory.

Modify the `mc-eval` procedure such that it checks whether the user typed in a list of length at least 2. If so, then mc-eval should return the second element of the list. Otherwise, mc-eval should just return what the user typed.

6. Copy the file ~`cs61a/lib/mceval-step2.scm` to your home directory.

Answer the following question: On line 17 with `((primitive-proc? exp) ...)`, we used `eval` On line 19 with `(map mc-eval (cdr exp))`, we used `mc-eval`

Why not use `eval` for line 19?

7. Copy the file ~`cs61a/lib/mceval-step3.scm` to your home directory.

Often times in Scheme, we would like to remove a variable name from the world completely. Implement a special form `(remove! var)` in our new Scheme such that it removes the variable and the value associated from the global environment list.