Name: ______________________________ Login: `cs61a-` ___

QUIZ #4

1. (0.5 points) Louis Reasoner doesn't understand why the **lazy evaluator** needs to use a special underlying representation for **thunks**. He suggests that thunks be implemented using **derived** expressions that create meta-Scheme procedures.

   In other words, Louis's modifications would cause expressions of the form:

   `(`*<compound-procedure>* *<arg$_1$>* ... *<arg$_n$>*`)`

   to be rewritten automatically as:

   `(`*<compound-procedure>* `(lambda ()` *<arg$_1$>*`)` ... `(lambda ()` *<arg$_n$>*`))`

   Louis rewrites `actual-value` appropriately to invoke these new thunks:

```
(define (actual-value exp env)
  (mc-eval (list exp) env))
```

   Even if Louis successfully integrates these changes into the lazy evaluator, why will his plan fail? Ignore details of memoization.

   On page 404 of the *SICP* textbook, Abelson and Sussman state:

   > A thunk must package an expression together with the environment, so that the argument can be produced later.

   Explain. Why do thunks need to store an environment? Give sample code that demonstrates this need.

Name: ______________________________ Login: `cs61a-` ___

2. (0.5 points) Write a procedure `depth` that calculates the maximum depth of a tree that is represented as a list. For example:

   ```
   (depth '())
   ```

   should return 0,

   ```
   (depth '(a b c d e))
   ```

   should return 1, and

   ```
   (depth '(a ((b) c) d (e)))
   ```

   should return 3.

3. (0.5 points) Which of the following *must* be **special forms**? Circle your answers.

   ```
   +                                       eval
   accumulate                              force
   apply                                   if
   begin                                   lambda
   car                                     let
   cdr                                     quote
   cond                                    set!
   cons                                    set-car!
   cons-stream                             set-cdr!
   define                                  stream-car
   delay                                   stream-cdr
   eq?
   ```

   What does `begin` do?