CS61A – Lab Assignment 1.2 | **Kurt Meinz**
University of California, Berkeley | Summer 2005

1. For each expression, give a definition of `f` such that evaluating the expression will not cause an error, and say what the expression's value will be, given your definition.

```
a. f            d. ((f))
b. (f)          e. (((f)) 3)
c. (f 3)
```

2. Find the values of the expressions

```
a. ((t 1+) 0)          b. ((t (t 1+)) 0)          c. (((t t) 1+) 0)
```

where `1+` is a primitive procedure that adds 1 to its argument, and `t` is defined as follows:

```
(define (t f)
  (lambda (x) (f (f (f x)))) )
```

Work this out yourself before you try it on the computer!

3. Find the values of the expressions

```
a. ((t s) 0)           b. ((t (t s)) 0)           c.  (((t t) s) 0)
```

where `t` is defined as in question 2 above, and `s` is defined as follows:

```
(define (s x)
  (+ 1 x))
```

4. Write a procedure `substitute` that takes three arguments: a *new* word, an *old* word, and a sentence. It should return a copy of the sentence, but with every occurrence of the old word replaced by the new word.

```
> (substitute 'maybe 'yeah '(she loves you yeah yeah yeah))
(she loves you maybe maybe maybe)
```

5. First, type the definitions

```
(define a 7)
(define b 6)
```

into Scheme. Then, fill in the blank in the code below with an expression whose value depends on both `a` and `b` to determine a return value of 24. Verify in Scheme that the desired value is obtained.

```
(let
  ((a 3) (b (+ a 2)))
  _____________ )
```

6. Write and test the `make-tester` procedure. Given a word `w` as argument, `make-tester` returns a procedure of one argument `x` that returns true if `x` is equal to `w` and false otherwise. Examples:

```
> ((make-tester 'hal) 'hal)
#t
> ((make-tester 'hal) 'cs61a)
#f
> (define sicp-author-and-astronomer? (make-tester 'gerry))
> (sicp-author-and-astronomer? 'hal)
#f
> (sicp-author-and-astronomer? 'gerry)
#t
```