CS61A – Lab Assignment 5.1 **Kurt Meinz**
University of California, Berkeley Summer 2002

1. This lab activity consists of example programs for you to run in Scheme. Predict the result before you try each example. If you don't understand what Scheme actually does, ask for help! Don't waste your time by just typing this in without paying attention to the results.

```
(define (make-adder n)
  (lambda (x) (+ x n)))

(make-adder 3)

((make-adder 3) 5)

(define (f x) (make-adder 3))

(f 5)

(define g (make-adder 3))

(g 5)

(define (make-funny-adder n)
  (lambda (x)
    (if (equal? x 'new)
        (set! n (+ n 1))
        (+ x n))))

(define h (make-funny-adder 3))

(define j (make-funny-adder 7))

(h 5)

(h 5)

(h 'new)

(h 5)

(j 5)

(let ((a 3))
  (+ 5 a))

(let ((a 3))
  (lambda (x) (+ x a)))

((let ((a 3))
   (lambda (x) (+ x a)))
 5)
```

```
((lambda (x)
   (let ((a 3))
     (+ x a)))
 5)

(define k
  (let ((a 3))
    (lambda (x) (+ x a))))

(k 5)

(define m
  (lambda (x)
    (let ((a 3))
      (+ x a))))

(m 5)

(define p
  (let ((a 3))
    (lambda (x)
      (if (equal? x 'new)
          (set! a (+ a 1))
          (+ x a)))))

(p 5)

(p 5)

(p 'new)

(p 5)

(define r
  (lambda (x)
    (let ((a 3))
      (if (equal? x 'new)
          (set! a (+ a 1))
          (+ x a)))))

(r 5)

(r 5)

(r 'new)
```

Continued on next page...

Lab Assignment 5.1 continued:

```
(define s
  (let ((a 3))
    (lambda (msg)
      (cond ((equal? msg 'new)
             (lambda ()
               (set! a (+ a 1))))
            ((equal? msg 'add)
             (lambda (x) (+ x a)))
            (else (error "huh?"))))))

(s 'add)

(s 'add 5)

((s 'add) 5)

(s 'new)

((s 'add) 5)

((s 'new))

((s 'add) 5)
```

```
(r 5)

(define (ask obj msg . args)
  (apply (obj msg) args)))

(ask s 'add 5)

(ask s 'new)

(ask s 'add 5)

(define x 5)

(let ((x 10)
      (f (lambda (y) (+ x y))))
  (f 7))

(define x 5)
```

2. Exercise 3.12 of Abelson and Sussman.

3. Suppose that the following definitions have been provided.

`(define x (cons 1 3))` `(define y 2)` A CS 61A student, intending to change the value of `x` to a pair with `car` equal to 1 and `cdr` equal to 2, types the expression `(set! (cdr x) y)` instead of `(set-cdr! x y)` and gets an error. Explain why.

4a. Provide the arguments for the two `set-cdr!` operations in the blanks below to produce the indicated effect on `list1` and `list2`. Do not create any new pairs; just rearrange the pointers to the existing ones.

```
> (define list1 (list (list 'a) 'b))
list1
> (define list2 (list (list 'x) 'y))
list2
> (set-cdr! _______________ _______________ )
okay
> (set-cdr! _______________ _______________ )
okay
> list1
((a x b) b)
> list2
((x b) y)
```

4b. After filling in the blanks in the code above and producing the specified effect on `list1` and `list2`, draw a box-and-pointer diagram that explains the effect of evaluating the expression `(set-car! (cdr list1) (cadr list2))` .

5. Exercises 3.13 and 3.14 in Abelson and Sussman.