CS 61A Midterm #3 — April 15, 2009

Your name ______________________________________

login: cs61a–_______

Discussion section number _______

TA's name ______________________________________

This exam is worth 40 points, or about 13% of your total course grade. It includes two parts: The individual exam (this part) is worth 35 points, and the group exam (the other part you probably just finished) is worth 5 points. The individual part contains six substantive questions, plus the following:

Question 0 (1 point): Fill out this front page correctly and put your name and login correctly at the top of each of the following pages.

This booklet contains seven numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book, open notes exam.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

If you want to use procedures defined in the book or reader as part of your solution to a programming problem, you must cite the page number on which it is defined so we know what you think it does.

READ AND SIGN THIS:

I certify that my answers to this exam are all my own work, and that I have not discussed the exam questions or answers with anyone prior to taking this exam.

If I am taking this exam early, I certify that I shall not discuss the exam questions or answers with anyone until after the scheduled exam time.

____

0	/1
1	/6
2	/6
3	/10
4	/4
5	/8
total	/35

Question 1 (6 points):

What will the Scheme interpreter print in response to **the last expression** in each of the following sequences of expressions? Also, **draw a "box and pointer" diagram for the final result of each sequence of expressions.** If any expression results in an error, **circle the expression that gives the error message** and just write "ERROR"; you don't have to give the precise message. Hint: It'll be a lot easier if you draw the box and pointer diagram *first*!

```
(define x (list 1 2))
(define y (list 3 4))
(define z (append x y))
(set-car! (cdr x) 5)
(set-car! y 6)
z
```

```
(define x (list 1 2))
(define y (list 3 4))
(define z (cons x y))
(set-car! (cdr x) 5)
(set-car! y 6)
z
```

```
(define x (list 1 2))
(define y (list 3 4))
(define z (cons x y))
(set! x 5)
(set! y 6)
z
```

Your name ______________________________________ login cs61a–_______

Question 2 (6 points): The following is an implementation of an object class in plain Scheme:

```
(define make-foo
  (let ((a 3))
    (lambda (b)
      (let ((c 4))
        (define (d e)
          (+ a e))
        (define (f g)
          (if (eq? g 'h)
              d
              (error "huh?")))
        f))))
```

Indicate which symbol in the code above corresponds to each OOP concept. Note: One of the questions has two answers (two symbols that match the concept), and one of the symbols is the answer to two of the questions!

Class variable:	a	b	c	d	e	f	g	h
Instance variable:	a	b	c	d	e	f	g	h
Instantiation variable:	a	b	c	d	e	f	g	h
Message:	a	b	c	d	e	f	g	h
Method:	a	b	c	d	e	f	g	h
Dispatch procedure:	a	b	c	d	e	f	g	h
Method argument:	a	b	c	d	e	f	g	h
Instance:	a	b	c	d	e	f	g	h

Question 3 (10 points):

(a) Given a list in which each element is a list of length 2, turn it into an association list in which the first element of each sublist becomes a key and the second element becomes the associated value. For example:

```
> (define ls (list (list 1 2) (list 3 4) (list 5 6)))
> ls
((1 2) (3 4) (5 6))
> (twos->assoc ls)
((1 . 2) (3 . 4) (5 . 6))
```

Use list mutation only; do not allocate any new pairs!

(Hint: Is the problem asking you to change the individual elements of the list, or is it asking you to rearrange elements?)

This question continues on the next page.

Question 3 continued:

(b) Given a list in which each element is a list of length 2, turn it into a flat list in which the elements of the sublists become the elements of the big list. For example:

```
> (define ls (list (list 1 2) (list 3 4) (list 5 6)))
> ls
((1 2) (3 4) (5 6))
> (twos->flat ls)
(1 2 3 4 5 6)
```

Use list mutation only; do not allocate any new pairs!

You don't have to keep the first pair of the argument list as the first pair of the returned list.

(Hint: Is the problem asking you to change the individual elements of the list, or is it asking you to rearrange elements?)

Question 4 (4 points):

```
(define (foo vec)
  (define (help result index sum len)
    (if (= index len)
        (begin (vector-set! result index sum)
               result)
        (begin (vector-set! result index (vector-ref vec index))
               (help result
                     (+ index 1)
                     (+ sum (vector-ref vec index))
                     len))))
  (help (make-vector (+ (vector-length vec) 1))
        0
        0
        (vector-length vec)))
```

What does Scheme return in response to the following expression, given the procedure definition above?

```
> (foo '#(5 6 7 8))
```

Your name ____ login cs61a–________

Question 5 (8 points):

One common operation in audio signal processing is *smoothing* a signal – reducing sudden short changes in volume. A simple smoothing technique is to *average* several consecutive values.

Write a procedure `stream-average` that takes two arguments: a stream `s` of numbers, and a positive integer `n`. It should return a stream in which each element is the average of `n` consecutive elements of `s`. For example:

```
> (show-stream (stream-average (list->stream '(1 20 300 4000 50000)) 3))
(107 1440 18100)   ; This is 321/3, 4320/2, and 54300/3.
```

You may assume that the argument stream is an *infinite* stream (unlike the example above).

Hint: Write a helper `stream-sums` that takes the same arguments and adds up `n` elements, then use that to compute `stream-average`.