CS 61A Midterm #1 — September 24, 2008

Your name _____

login: cs61a–_____

Discussion section number _____

TA's name _____

This exam is worth 40 points, or about 13% of your total course grade. The exam contains 7 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 7 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 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 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
1–2	/9
3	/3
4	/6
5	/8
6	/6
7	/7
total	/40

Question 1 (5 points):

What will Scheme print in response to the following expressions? If an expression produces an error message, you may just write "error"; you don't have to provide the exact text of the message. If the value of an expression is a procedure, just write "procedure"; you don't have to show the form in which Scheme prints procedures.

```
(accumulate word (map (lambda (x) (remainder x 2))
                      '(1 2 3 4 5 6 7)))          _____

(or #f 3 #t)                                      _____

((lambda (x) (x x x)) 7)                          _____

(let ((+ *) (a (+ 2 3))) (+ a 10))                _____

(caaddr '((a b c d e) (f g h i j) (k l m n o))     _____
```

Question 2 (4 points):

What will Scheme print in response to the following expressions? If an expression produces an error message, you may just write "error"; you don't have to provide the exact text of the message. **Also, draw a box and pointer diagram for the value produced by each expression.**

```
(list (cons '(1) '(2)) (list '(1) '(2)))          _____
```

```
(cons (cons 1 2) (append '(1) '(2)))              _____
```

Your name _____ login cs61a–_____

Question 3 (3 points):

```
(define (foo x) 'done)
```

What is the result of `(foo (foo))` (yes, this is what we mean!)

a. in normal order

b. in applicative order

```
(define (inc x) (+ x 1))
```

c. True or False: In evaluating `(+ (inc 3) (inc 3))`, `+` gets called more in normal than in applicative order.

_____True _____False

Question 4 (6 points):

(a) What is the order of growth in time of `adding` below, in terms of n, the length of its argument? **(Note: the `count` procedure takes time $\Theta(n)$.)** Also, does `adding` generate an iterative or a recursive process?

```
(define (adding sent)
  (if (= (count sent) 0)
      0
      (+ (first sent) (adding (bf sent)))))
```

_____$\Theta(n)$ _____$\Theta(n^2)$ _____$\Theta(2^n)$ _____Not enough information to know

_____Iterative _____Recursive

(b) Consider the three procedures below:

```
(define (foo n temp)
   (if (= n 0) temp
       (foo (- n 1) (+ temp n))))

(define (bar n)
   (+ (foo n 0) (foo n 0)))

(define (baz n)
   (foo (foo (foo n 0) 0) 0))
```

Does procedure `foo` generate an iterative or a recursive process?

_____Iterative _____Recursive

What is the order of growth in time of procedure `foo`?

_____$\Theta(n)$ _____$\Theta(n^2)$ _____$\Theta(2^n)$ _____$\Theta(\log n)$

What is the order of growth in time of procedure `bar`?

_____$\Theta(n)$ _____$\Theta(n^2)$ _____$\Theta(2^n)$ _____$\Theta(\log n)$

What is the order of growth in time of procedure `baz`?

_____$\Theta(n)$ _____$\Theta(n^2)$ _____$\Theta(n^3)$ _____$\Theta(n^4)$

Your name _____ login cs61a–_____

Question 5 (8 points):

For this question (both parts), use only higher order procedures, not recursion, even in helper procedures!

(a) Write a procedure `make-map-adder` that, given a number, returns a function that will add that number to each number in a sentence of numbers.

Example call:

```
> ((make-map-adder 5) '(1 3 5 8))
(6 8 10 13)
```

(b) Now write a procedure `make-map-maker` that takes a *two-argument function* `fn` as its argument, and returns a function of one argument `arg` that returns a function that applies `fn` to `arg` and each word of a sentence. For example, `(make-map-maker +)` should return the function `make-map-adder` from part (a).

Your name _____ login cs61a–_____

Question 6 (6 points):

Note: Use recursion, not higher-order functions, to solve this problem.

Write a procedure `insert-multiples` that takes a sentence, a word, a position number, and a count. (The last two are nonnegative integers.) It should return the result of inserting the given word into the sentence, at the given position, as many times as count states. Assume the position is legal (i.e., it exists in the sentence). For example:

```
STk> (insert-multiples '(this is a sentence) 'balloon 2 4)
(this is balloon balloon balloon balloon a sentence)
```

Question 7 (7 points):

Alyssa P. Hacker has written the constructor and selectors for a three-piece data structure:

```
(define (make-game r p s)
  (list r p s))

(define rock car)
(define paper cadr)
(define scissors caddr)
```

(a) Now Ben Bitdiddle plays a trick on Alyssa by redefining `paper`:

```
(define paper cdar)
```

Help Alyssa by redefining as many of the other three procedures as necessary so that they work correctly with Ben's new version of `paper`.

(b) Each of the three components of a game (rock, paper, and scissors) is a sentence. Alyssa has hired Louis Reasoner to write a procedure that extracts the second word of each sentence and puts the result in a sentence. Unfortunately, Louis doesn't believe in data abstraction, and he didn't know about Ben's change, so he wrote the following, based on Alyssa's original definition of the constructor and selectors:

```
(define (seconds game)
  (list (cadar game) (cadadr game) (cadaddr game)))
```

Help Louis by rewriting his procedure to respect the data abstractions.