CS 61A Midterm #1 — September 23, 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. 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 6 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	/6
3–4	/6
5	/9
6	/12
7	/6
total	/40

Question 1 (2 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.

```
(every (lambda (x) (se x x))
       (keep (lambda (x) (even? (count x)))
             '(and your bird can sing)))                ____________________

((lambda (x y) (x (y 3)))
 (lambda (x) (* x x))
 (lambda (x) (+ x 6)))                                  ____________________
```

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 (list (cons 3 (list 4))))                         ____________________
```

```
(append (list 1 2) (list 4 (cons 2 3)))                 ____________________
```

Your name ____ login cs61a–_______

Question 3 (2 points):

```
(define (square x)
  (* x x))

(square (+ 2 3))
```

How many times is + called in:

Normal order ______________________ Applicative order ______________________

Question 4 (4 points):

What is the order of growth in time of each of the following procedures, in terms of their argument value n? Also, does each generate an iterative process or a recursive process?

```
(define (mystery n)
  (cond ((< n 0) 0)
        ((odd? n) (+ 2 (mystery (- n 2))))
        (else (+ 1 (mystery (- n 1))))))
```

______$\Theta(1)$ ______$\Theta(n)$ ______$\Theta(n^2)$ ______$\Theta(2^n)$

______Iterative ______Recursive

```
(define (bar n)
  (if (or (> n 10) (< n 0))
      n
      (bar (+ n 1))))
```

______$\Theta(1)$ ______$\Theta(n)$ ______$\Theta(n^2)$ ______$\Theta(2^n)$

______Iterative ______Recursive

Question 5 (9 points):

Eight TAs are trying to write a midterm. Brian decides that the problems should be represented using his "problem" ADT, which uses the constructor provided below:

```
(define (make-problem question solution points)
  (list (list question solution) points))
```

(a) Write selectors for this ADT.

(b) The exam has to not be worth too many or too few points. Write a procedure `total-points` which takes a list of problems as its argument, and returns the sum of their point values.

(c) Brian decides to add an `expected-time` attribute to problems by using the following new constructor:

```
(define (make-problem question solution points expected-time)
  (list (list question solution) expected-time points))
```

Assuming the selectors are changed accordingly, what else, if anything, would you need to change to make your answer in part (b) still work?

Question 6 (12 points):

You are going to write two versions of a function appearances that takes two arguments, a sentence and a word, and returns the number of occurrences of the word in the sentence:

```
> (appearances '(I love cs61a just like I love oranges) 'love)
2

> (appearances '(I love cs61a just like I love oranges) 'oranges)
1

> (appearances '() 'test)
0
```

(a) Write a version of appearances **using only recursion.** Do **not** use any higher-order functions!

(b) Now write a version of appearances **using only higher-order functions.** Do **not** use recursion!

Question 7 (6 points):

Write a procedure `do-n` that takes three arguments: a function `func` of one argument, a list `args` of data, and a list `times` of numbers. Your procedure will take each element of `args`, and apply `func` to that element, then again to the result, then to *that* result, etc., repeating the use of `func` the number of times given in the corresponding element of the list `times`. You may assume the two lists are the same length. You may further assume the value returned by `func` will be a valid argument to that function.

You may write and use helper procedures.

For example:

```
> (do-n square '(1 2 3) '(9 3 2))  ;2nd number in result is 256 because
(1 256 81)                         ; 2^2 = 4;  4^2 = 16;  16^2 = 256

> (do-n butlast '(twinkle twinkle little star) '(2 5 4 1))
(twink tw li sta)

> (do-n cdr '((mary had a little lamb) (with fleece as white as snow))
            '(2 3))
((a little lamb) (white as snow))
```

Start with this:

```
(define (do-n func args times)
```