CS 61A Midterm #1 — February 16, 2005

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 6 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 especially difficult, leave it for later; start with the ones you find easier.

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	/7
3–4	/10
5	/8
6	/8
total	/40

Note: Unlike the version in the lecture notes, the primitive `keep` function includes words as well as sentences in its domain and range. This note applies to the entire exam, not just this page.

Question 1 (6 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.

```
(butfirst '(yesterday))                                   ________________
```

```
(- (+) (*))                                               ________________
```

```
((lambda (x y) (word y x)) '(abcd efgh))                  ________________
```

```
(keep (member? letter 'aeiou) 'redirection)               ________________
```

```
(let ((a 'b) (b 'a)) (word a b b a))                      ________________
```

```
((lambda (w) (equal? (first w) (last w))) 'kink)          ________________
```

Your name _____ _________________ login cs61a–_______

Question 2 (7 points):

Write the procedure `devowel` that takes a sentence as its argument, and returns a sentence in which every vowel has been removed from the argument. For example:

```
> (devowel '(love me do))
(lv m d)
```

Use higher-order functions, not recursion.

You may assume that you are given `vowel?` and `consonant?` predicates.

Question 3 (4 points):

Procedure `min` below takes a nonempty sentence of numbers as its argument, and returns the smallest number in the sentence.

```
(define (min s)
  (define (helper s min-so-far)
    (cond ((empty? s) min-so-far)
          ((< (first s) min-so-far) (helper (bf s) (first s)))
          (else (helper (bf s) min-so-far))))
  (helper (bf s) (first s)))
```

What is its running time, in terms of N, the size of s?

_____$\Theta(1)$ _____$\Theta(N)$ _____$\Theta(N^2)$ _____$\Theta(2^N)$

How much space does it require?

_____$\Theta(1)$ _____$\Theta(N)$ _____$\Theta(N^2)$ _____$\Theta(2^N)$

Question 4 (6 points):

Suppose you've defined the following procedure:

```
(define (silly-func wd)
  (se (bf wd) (bl wd) (word wd wd)))
```

How many times is the * procedure invoked during the evaluation of the following expression:

```
(silly-func (* 3 4))
```

(a) in applicative order? ____________________

(b) in normal order? ____________________

(c) in actual Scheme? ____________________

Your name _______ _______________ login cs61a–______

Question 5 (8 points):

Write a function `count-zero-crossings` that takes a sentence of numbers as its argument. It should return the number of "zero crossings" in the sentence; a zero crossing occurs when two consecutive nonzero numbers, or two nonzero numbers separated only by zeros, have opposite signs.

```
> (count-zero-crossings '(5 3 1 -1 -3 2 0 4))
2

> (count-zero-crossings '(3 1 2 0 0 -4 0 -2))
1
```

Use recursion, not higher order functions.

Your name ____ login cs61a–_______

Question 6 (8 points):

Write a function `make-splitter` that takes a predicate `pred` as its argument. The domain of `pred` will include words. `Make-splitter` should return a function whose argument is a word, and whose return value is a two-word sentence, in which the first word contains the letters of the argument for which `pred` returns true, and the second word contains the letters for which `pred` returns false. For example:

```
> (define vowels (make-splitter vowel?))
> (vowels 'catastrophe)
(aaoe ctstrph)

> ((make-splitter number?) '1after909)
(1909 after)
```