CS 61A, Summer 2006, Midterm 1 **July 14, 2006**
Instructor: Kevin Lin

READ EVERYTHING BEFORE PROCEEDING: This exam is worth 40 points, or about 13% of your total course grade. You have two hours (120 minutes) to complete the exam. You may not use any books, electronic devices, the help of your peers, etc. You may use exactly one standard size page of notes (front and back).

When writing procedures in this exam, don't put in error checks. You may assume that you will be given arguments of the correct type.

Make good use of your time. If you find one question especially difficult, leave it for later; start with the ones you find easier.

The exam contains 6 substantive questions (one page per question – check to see that you are not missing any pages in your exam), plus the following question:

QUESTION 0 (1 point)
Fill out all of the following information correctly:

Your name:

Your login:

Your SID:

Your TA's name:

READ THIS AND SIGN BELOW: 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.

Sign here:

Now put your name and login at the top of each page in this packet.

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 error message. If the value of the expression is a procedure, just write "procedure"; you don't have to show the form in which Scheme prints procedures.

```
(se (last '(cal)) (butlast '(golden)) '(bears))
```

```
(every (lambda (x) (keep #t x)) '(hi there))
```

```
((lambda (x) (x 5)) (lambda (y) (+ y 2)))
```

```
(if 'kevin 'gap ramesh)
```

```
(let ((bf first) (first bf))
  (word (bf 'zidane) (first 'redcard)))
```

```
(lambda (x) *)
```

QUESTION 2 (4 points)
Consider the procedure below:

```
(define (baz n)
  (define (square x) (* x x))
  (define (helper x)
    (cond ((= x (square n)) 17)
          ((even? x) (+ 4 (helper (+ x 1))))
          (else (helper (+ x 1)))))
  (helper 0))
```

What is the order of growth in time of `baz`?
Circle one: $\Theta(1)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(2^n)$.

What is the order of growth in space of `baz`?
Circle one: $\Theta(1)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(2^n)$.

QUESTION 3 (4 points)
Write a procedure `negevens` that takes a sentence of numbers as argument and returns a sentence of the negatives of the even numbers. For example:

```
> (negevens '(-1 -4 5 6 0 2 7))
(4 -6 0 -2)
```

Use higher order functions. Do not use recursion!

QUESTION 4 (7 points)
Suppose we want to implement a database with entries that represent students in a CS class. We want each entry to store a student's first name, last name, student ID number, homework grade, project grade, and exam grade. Our constructor function is implemented as follows:

```
(define (make-student-entry sid         ; an 8-digit number
                            first-name  ; a word
                            last-name   ; a word
                            hw-grade    ; an integer between 0 and 100
                            proj-grade  ; an integer between 0 and 100
                            exam-grade) ; an integer between 0 and 100
  (list sid
        (cons first-name last-name)
        (list hw-grade proj-grade exam-grade)))
```

Draw the box-and-pointer diagram corresponding to the following expression.

```
(make-student-entry 12345678 'john 'doe 75 85 95)
```

Fill in the blanks for the selectors `last-name` and `exam-grade`:

```
(define (last-name student-entry)
; returns the last name of a student


)

(define (exam-grade student-entry)
; returns a student's exam grade


)
```

QUESTION 5 (9 points)
Write a procedure `count-big-jumps` that takes a sentence of numbers, and a jump number. It returns the number of times that absolute value of the difference between two adjacent numbers in the sentence is greater than the jump number. For example:

```
> (count-big-jumps '(1 0 4 8 9 0 6) 2)
4
> (count-big-jumps '(2 4 6 8 10) 3)
0
```

Use recursion. Do not use higher order functions! You may use the procedure `abs`, which takes a number and returns its absolute value.

QUESTION 6 (9 points)
Write a procedure `first-bf-func`. It takes as argument a sentence containing the *words* `first` and `bf` and returns a procedure of one argument that applies the `first` and `bf` *procedures* to its argument in the opposite order that the corresponding *words* appear in the sentence. For example:

```
> ((first-bf-func '(first bf)) 'computerscience)
o
> ((first-bf-func '(bf bf first)) '(italy france))
aly
> ((first-bf-func '(bf bf)) '(homer marge bart lisa maggie))
(bart lisa maggie)
```