Your name ______________________________________

TA's name _________________________ Discussion section number ______

A random five-digit number: ____________

Circle the last two letters of your login (`cs61a-xx`)

```
a b c d e f g h i j k l m n o p q r s t u v w x y z 1 2 3

a b c d e f g h i j k l m n o p q r s t u v w x y z
```

This exam is worth 40 points, or about 13% of your total course grade. The exam contains eight substantive questions, plus the following:

Question 0 (1 point): Fill out this front page correctly and correctly copy your random five-digit number to the top of each of the following pages. (This is to make sure the pages of your exam stay together even if the staple comes out.)

This booklet contains eight 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 a 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
1-2	/12
3-4	/7
5	/4
6	/4
7	/6
8	/6
total	/40

Question 1 (5 points):

What will Scheme print in response to the following expressions? If the expression produces an error message, you may just write "error"; you don't have to provide the exact text of the message. If the expression results in an infinite loop, you should write "infinite loop".

```
> (define (poof x)
    (lambda (y) (+ x y)) )
> (define bam 10)
> ((poof bam) 4)
```

____

```
> (define swoosh
    (lambda (num)
      (lambda (y) (+ y num)) ))
> (define kablooie 10)
> ((swoosh kablooie) 4)
```

____

```
> (define (mystery sent)
    (if (empty? sent)
        '()
        (se (first sent) (mystery sent)) ))
> (mystery '(you cant always get what you want))
```

____

```
> (define garply 5)
> (and (> garply 0) (/ 20 garply))
```

____

```
> (define brrrm '((we need to) (go (deeper))))
> (cdar brrrm)
```

____

Your five-digit number: ______________

Question 2 (7 points):

Fill in the blanks with a name of the procedure so that the result will be correct. Then, draw the box-and-pointer diagrams for each result.

```
> (______________ 'a '(b . c) #f)
(a (b . c) #f)
```

```
> (______________ '(a) '(b))
((a) b)
```

```
> (______________ (______________ #t #f) '(maybe dunno))
(#t #f maybe dunno)
```

Question 3 (4 points):

What is the order of growth of the following procedures?

```
(define (bar n)
  (define (baz n)
    (if (= n 0)
        0
        (+ n (baz (- n 1)))))
  (* (baz n) (baz n)))
```

The running time of `bar` is $\Theta($_______$)$

```
(define (foo n)
  (if (= (remainder n 7) 0)
      n
      (foo (- n 1))))
```

The running time of `foo` is $\Theta($_______$)$

Question 4 (3 points):

(For reference, here's the implementation of `count`*.)*

```
(define (count sent)
  (if (empty? sent)
      0
      (+ 1 (count (bf sent))) ))
```

Let's say we have a list of n sentences called `paragraph`. Each sentence has n words. If we run `(map count paragraph)`, what's the order of growth of the running time of this call to `map`?

$\Theta(1)$ _______ $\Theta(n)$ _______ $\Theta(n^2)$ _______ $\Theta(n^3)$ _______ $\Theta(2^n)$ _______

Now let's say we run `(map selection-sort paragraph)`. (`selection-sort` is a $\Theta(n^2)$ procedure.) What's the order of growth of the running time of this call to `map`?

$\Theta(1)$ _______ $\Theta(n)$ _______ $\Theta(n^2)$ _______ $\Theta(n^3)$ _______ $\Theta(2^n)$ _______

In general, if it takes time $t(n)$ to run a procedure `f` on an input of length n, how much time does it take to map `f` over a list of n such inputs?

$\Theta($_______$)$

Your five-digit number: ______________

Question 5 (4 points):

Does `squares` generate a iterative process or recursive process?

```
(define (squares nums)
  (if (empty? nums)
      '()
      (se (square (first nums)) (squares (bf nums)))
```

Iterative _______ Recursive _______

Now, rewrite `squares` recursively (if you said "iterative") or iteratively (if you said "recursive").

```
(define (squares nums)
```

Question 6 (4 points):

Say we have a list storing people's ages:

```
> up-ages
((kevin 24) (fredrickson 76) (dug 7) (ellie 77))
```

Here, Kevin is 24 years old (represented by the list `(kevin 24)`), and Ellie is 77 years old.

Write a procedure `separate` that separates the names into one side of the list, and the ages into another side.

```
> (separate '((kevin 24) (fredrickson 76) (dug 7) (ellie 77)))
(kevin fredrickson dug ellie 24 76 7 77)
```

For this problem, use higher-order procedures. **Do not use recursion!**

```
(define (separate lst)
```

Question 7 (6 points):

Write a procedure `doubles` that takes a sentence and returns a new sentence with each word doubled. The trick is that numbers are doubled numerically while other words are repeated.

```
> (doubles '(1 2 abc 3 4))
(2 4 abcabc 6 8)
> (doubles '(its over 9000 !))
(itsits overover 18000 !!)
```

You may use the primitive `number?` predicate. For this problem, **do not use higher-order procedures!**

```
(define (doubles sent)
```

Question 8 (6 points):

You are lost on the moon again, this time without your team and all the cool equipment. You only have an oxygen tank, your suit, and your computer. The directions back to your base were corrupted and some random numbers were mixed into the sentence. Fortunately, your computer was able to make sure no actual numbers were corrupted, only words that are made up of letters.

Write a function `recover` to decode a sentence corrupted in this way.

```
> (recover '(g123123213o e98734ast for123 100 99ste9ps))
'(go east for 100 steps)
```

You may use the primitive `number?` predicate. Remember, the fourth word of the example sentence is `100` in the result, not the empty word!

```
(define (recover sent)
```