CS61A, Spring 2003
Midterm #1
Professor Brian Harvey

Question 1 (7 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.

```
(word (first 'happy) (last '(we are)))
```

```
(((lambda (a b) b) + *) 4 5)
```

```
((first '(+ *)) 4 5)
```

```
(every last '(blackboards eat chalk))
```

```
(every member? '(t e the))
```

```
(keep (< 6) '(2 8 1 30))
```

```
butfirst
```

Question 2 (8 points):

Write the procedure *powers* that behaves as shown here:

```
> (powers 4)
(1 2 4 3 9 27 4 16 64 256)
```

(The underlining isn't part of the actual result; it's just there to help you understand the result.) *Powers* takes a positive integer argument n. It returns a sentence containing, for every k from 1 to n, all the powers of k up to k^k. So the result above is

(1^1 2^1 2^2 3^1 3^2 3^3 4^1 4^2 4^3 4^4)

Hint: Use a helper that generates the powers of a single value of k.

Question 3 (8 points):

We are going generalize the idea of

```
(define (plural wd)
  (word wd 's))
```

by allowing prefix and/or suffix to be added to a word. Here are some examples:

```
> (define plural (word-maker '(* s)))
> (plural 'book)
books


> (define past (word-maker '(* ed)))
> (past 'walk)
walked


> ((word-maker '(re *)) 'write)
rewrite


> ((word-maker '(de * ing)) 'construct)
deconstructing


> ((word-maker '(* -vs- *)) 'spy)
spy-vs-spy
```

Word-maker takes a sentence, called a *template*, as its argument. It returns a function that takes a word as argument and returns another word based on the template, but with any * in the template replaced with the argument word and the result scrunched into a word. Write word-maker.

Don't use recursion; use higher-order functions. Here's a helper function for you:

```
(define (scrunch sent)
  (if (empty? sent)
    ""
    (word (first sent) (scrunch (butfirst sent)))))


> (scrunch '(here there and everywhere))
herethereandeverywhere
```

Question 4 (4 points):

The function below, truncate, takes a sentence and cuts it off at the first occurrence of the word "end". If that word never appears, it returns the whole sentence.

```
(define (truncate sent)
  (define (helper part1 part2)
    (cond ((empty? part2) part1)
      ((eq? (first part2) 'end) part1)
      (else (helper (se part1 (first part2)) (bf part2)))))
  (helper '() sent))
```

What is the most useful invariant of the helper function helper?

Question 5 (2 points):

What's the order of growth of the procedure nine-times defined below?

```
(define (nine-times num)
  (three-times (three-times num)))


(define (three-times a)
  (if (= a 0)
    0
    (+ 3 (three-times (- a 1)))))
```

Question 6 (6 points):

```
(define (count-to-1000000 n)
  (if (= n 1000000)
    n
    (count-to-1000000 (+ n 1))))
```

Which of the function calls below would you expect to run faster in normal order, and which in applicative order, assuming that count-to-1000000 takes the same amount of time in both?

```
(define (foo a)                                   Circle one:
  (if (= 1 2)                                     normal-faster  applic-faster  same
    a
    '(one does not equal two)))
> (foo (count-to-1000000 1))

(define (bar a) (+ a a))                          Circle one:
> (bar (count-to-1000000 1))                      normal-faster  applic-faster  same

(define (baz a b) (+ a b))                        Circle one:
> (baz (count-to-1000000 1) (count-to-1000000 1)) normal-faster  applic-faster  same
```

Question 7 (4 points):

I'm trying to write a function that determines whether a word is a palindrome (a word that is spelled the same forward and backward, such as "stats"). However, I'm having some problems with my function. So I've been running several test cases for my program, as follows:

```
STk> (palindrome? 'foot)      STk> (palindrome? "")
#f                            #t
STk> (palindrome? 'toot)      STk> (palindrome? 123421)
#t                            #f
STk> (palindrome? 'stars)     STk> (palindrome? 'abcddcba)
#f                            #t
STk> (palindrome? 123321)     STk> (palindrome? 123)
#t                            #f
STk> (palindrome? 'stats)     STk> (palindrome? 'abcdcba)
error                         error
```

Based on these test results, what is the error in my program?