CS61A, Spring 2001
Midterm #1
Professor Brian Harvey

Question 1 (5 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 (butlast (se '(this) 'is '(easy) )))
```

```
(car ((lambda (lst) (cdr lst)) '((1 2) (3 4)) ) )
```

```
(let ((rotate (lambda (a b c)
               (if (number? a)
                   a
                   (b c 2 3)))))
  (rotate rotate rotate 1))
```

```
(if (equal? '2 2) + -)
```

```
(map butfirst
     '((she loves you) (help!) (penny lane)))
```

Question 2 (6 points):

The following program takes two arguments: a number and a *list of sentences.* It returns a *sentence* containing the *n*th word (counting from zero) from each of the sentences in the argument list:

```
> (every-nth 2 '((a b c d) (e f g h)))
(c g)
```

Fill in the blanks, **respecting the data abstraction!**

```
(define (every-nth num list-of-sents)
  (define (nth num sent)
    (if (= num 0)
        ( __________ sent)
        (nth (- num 1) ( __________ sent))))
  (if ( __________ list-of-sents)
      '()
      ( __________ (nth num ( __________ list-of-sents))
                  (every-nth num ( __________ list-of-sents))
```

Question 3 (6 points):

```
(define (enumerate-buzz start end)
  (if (> start end)
      '()
      (cons (buzz start)
            (enumerate-buzz (+ start 1) end))))


(define (buzz x)
  (if (or (member? 7 x) (equal? (remainder x 7) 0))
      'buzz
      x))
```

(a) What does (**enumerate-buzz** 10 20) return?

(b) Assuming that all the primitive procedures used take constant time, indicate the order of growth in time of **enumerate-buzz**: Theta (_____)

(c) What kind of process does **enumerate-buzz** generate? ___Recursive ___Iterative

Question 4 (6 points):

Given the following definition:

```
(define (garply x)
   (+ (* x x) 1))
```

...and expression:

```
(garply ( (lambda (y) (* y 2)) (* 2 2) ) )
```

(a) How many calls to the * (multiplication) operator will there be

in normal order? ________

in applicative order? ________

(b) What does the expression return

in normal order? ________

in applicative order? ________

Question 5 (8 points):

This question concerns the twenty-one game used in the first programming project.
(Assume the version without jokers.)

(a) Write a procedure **contrary** that takes a *strategy* as its argument, and returns the opposite strategy. That is, if the input strategy would "hit", **contrary** would return a strategy that doesn't hit, and vice-versa.

(b) Assume that the following variables are already defined: **my-strategy**, **my-hand**, and **his-card**, representing a particular strategy, customer's hand and dealer's card. Write an expression using **contrary** that would return whether the contrary strategy would want another card in that situation. You may not use the function **not** or the special forms **and**, **or**, **if**, and **cond** in this expression.

(c) Write a procedure **suit-map** that takes two arguments: a function and a hand. It should return a *list* of four elements, each of which is the result of applying the function to the cards from the hand that are of a particular suit, in the order diamonds, clubs, hearts, spades. Examples:

```
> (suit-map count '(3S 4D JS 8C 3D))
(2 1 0 2)
> (suit-map empty? '(3S 4D JS 8C 3D))
(#f #f #t #f)
> (suit-map (lambda (x) x) '(3S 4D JS 8C 3D))
((4D 3D) (8C) () (3S JS))
```

Question 6 (8 points):

Mad Libs (tm) is a campfire game in which players fill in blank places in a story, thereby creating a hilarious new story. We wish to implement this in Scheme.

Mad-libs is a procedure that takes a *sentence* **story** as its argument. Some of the words in **story** begin with '*', meaning they are to be replaced. **Mad-libs** returns a procedure that takes *two* sentences, **nouns** and **adjectives**, and replaces, in order, words from **story** beginning with '*' with words from nouns if the word is *NOUN or from **adjectives** if the word is *ADJECTIVE. For example, an infamous author once wrote:

It was a dark and stormy night... and the rain fell in torrents - except at occasional intervals, when it was checked by a violent gust of wind which swept up the streets (for it is in London that our scene lies), rattling along the housetops, and fiercely agitating the scanty flame of the lamps that struggled against the darkness. - Paul Clifford (1830)

```
(define my-story
  (mad-libs '(It was a *ADJECTIVE and *ADJECTIVE *NOUN)))

> (my-story '(night) '(dark stormy))
(It was a dark and stormy night)

> (my-story '(midterm) '(fun easy))
(It was a fun and easy midterm)
```

Write ***mad-libs****. Use the proper selectors and constructors for words and sentences. You may assume that the lengths of* ***nouns*** *and of* ***adjectives*** *are exactly the same as the number of words of that category in* ***story****.*