Sample midterm 2 #2

Problem 1 (What will Scheme print?)

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. **Also, draw a box and pointer diagram for the value produced by each expression.**

```
(list (cons 2 3) 4)
```

```
(append (cons '(a) '(b)) '(c))
```

```
(cdadar '((e (f) g) h))
```

Problem 2 (Backwards what-will-Scheme-print.)

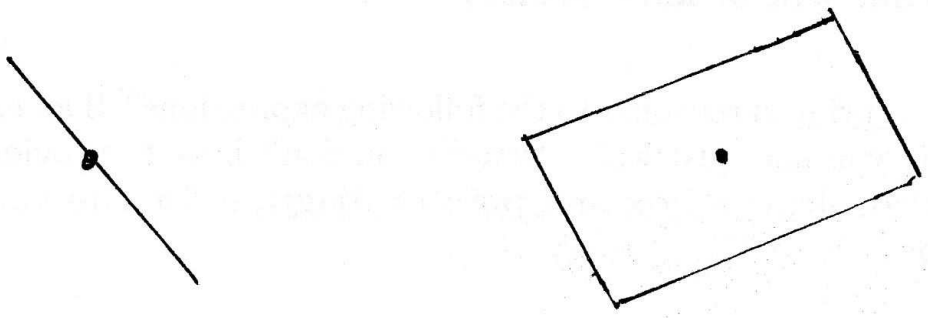
Fill in the blank in this expression:

(_____ '(G (H I) J))

so that the value of the expression is the letter `I`.

Problem 3 (Project 2 data abstraction)

This question concerns the picture project. The *midpoint* of a segment is the point halfway between its two ends. The midpoint of a frame is the point in that frame corresponding to the point (0.5, 0.5) in the unit square:



We want a procedure named `midpoint` that takes either a segment or a frame as argument, and returns the vector from the origin to the midpoint of the argument. We'll accomplish this in stages.

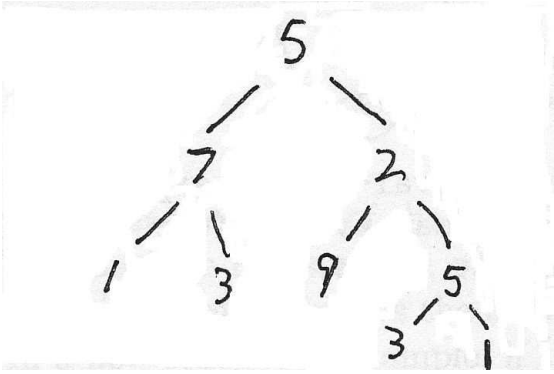
(a) Write new versions of the constructor `make-segment` and the selectors `start-segment` and `end-segment` that include the word `segment` as a type tag in the internal representation.

(b) We'll assume that the constructor and selectors for frames have been modified similarly so that the word `frame` is included as a type tag in frames; you need not write these procedures. Now write the procedure `midpoint` that takes either a segment or a frame as its argument, and returns the vector from the origin to the midpoint of the argument. **Respect all relevant data abstractions.** If the argument is neither a segment nor a frame, your procedure should return `#f`.

Problem 4 (Tree recursion)

This question concerns Trees, using the constructor `make-tree` and the selectors `datum` and `children` as discussed in lecture.

Suppose we are dealing with Trees in which the datum at every node is a number. Write a procedure `maxpath` that takes such a Tree as its argument, and returns the largest possible sum of numbers along a path from the root node to some leaf node. For example, in the Tree



the largest such sum is $5 + 2 + 9$, so the procedure should return 16.

Respect the data abstraction!

You may use, without defining it, the procedure `biggest` that takes a nonempty list of numbers as its argument and returns the largest number in the list:

```
> (biggest '(10 23 7 5))
23
```

Problem 5 (Scheme-1 interpreter)

For reference, here are the central procedures of `scheme-1`, with the lines numbered:

```
 1  (define (eval-1 exp)
 2    (cond ((constant? exp) exp)
 3          ((symbol? exp) (eval exp))
 4          ((quote-exp? exp) (cadr exp))
 5          ((if-exp? exp)
 6           (if (eval-1 (cadr exp))
 7               (eval-1 (caddr exp))
 8               (eval-1 (cadddr exp))))
 9          ((lambda-exp? exp) exp)
10          ((pair? exp) (apply-1 (eval-1 (car exp))
11                                (map eval-1 (cdr exp))))
12          (else (error "bad expr: " exp))))

13  (define (apply-1 proc args)
14    (cond ((procedure? proc)
15           (apply proc args))
16          ((lambda-exp? proc)
17           (eval-1 (substitute (caddr proc)
18                               (cadr proc)
19                               args
20                               '())))
21          (else (error "bad proc: " proc))))
```

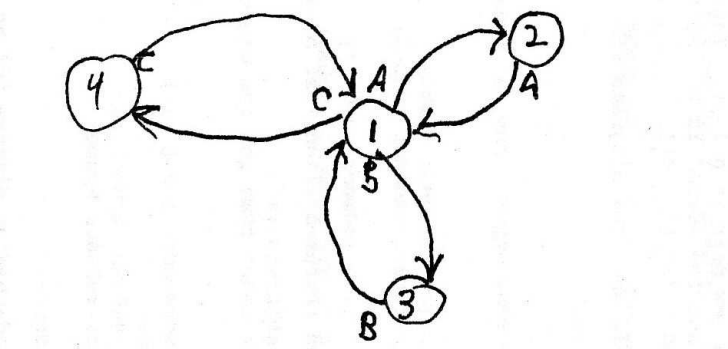
A student tries to type this into his computer, but makes one mistake. Here is a transcript of some of his test cases:

```
Scheme-1: (+ 2 3)
5

Scheme-1: (+ (* 2 2) 3)
ERROR

Scheme-1: ((lambda (x) (* x x)) 2)
4

Scheme-1: ((lambda (x) (* x x)) (+ 1 1))
ERROR

Scheme-1: (if #t 2 3)
2

Scheme-1: (if (> 3 0) (+ 2 1) (+ 3 1))
3
```

Based on the test cases, what is wrong with his version of `scheme-1`? Indicate the line number with the problem, and what the student typed on that line.

Problem 6 (Data directed programming)

This two-part question is about data-directed programming. We are going to use a list to represent a *finite state machine* (FSM) like this:



The numbered circles represent the *states* of the machine. At any moment the machine is in a particular state. The machine reads words, one letter at a time. If the machine is in some state, and sees a certain letter, it follows the arrow from its state with that letter as its label, and ends up in a new state. For example, if the machine above is in state 1 and it sees the letter B, it goes into state 3.

We'll represent a FSM by a list of all its transition arrows. The machine above has six such arrows:

```
((1 A 2) (1 B 3) (1 C 4) (2 A 1) (3 B 1) (4 C 1))
```

If the machine reads a letter for which there is no applicable arrow, it should enter state number 0. In effect, there are "invisible" arrows like `(2 C 0)` that needn't be represented explicitly in the list.

(a) Write a function `transition` that takes three arguments: a FSM list, a current state number, and a letter. The function should return the new state. For example, if `fsm` represents the list above, we should be able to do this:

```
> (transition fsm 1 'C)
4
```

```
> (transition fsm 2 'C)
0
```

(b) We want an FSM to process words, not just single letters. The machine "reads" the word one letter at a time. Our sample FSM, starting in state 1, should process the word AACCAAB by going through states 2, 1, 4, 1, 2, 1, and 3. The final state, 3, is the result of processing the word.

Write a function `process` that takes as its arguments a FSM, a starting state number, and a word. It should return the ending state:

```
> (process fsm 1 'AACCAAB)
3
```

```
> (process fsm 1 'AAAC)
0
```

Problem 7 (Object oriented programming)

(a) Check the cases in which the first class shown should be the PARENT of the other class:

_____cell phone / keypad

_____building / office building

_____stapler / staple

_____arm / arm bone

_____bone / arm bone

_____person / arm bone

(b) For each of the following indicate CLASS VARIABLE or INSTANCE VARIABLE:

_____CLASS _____INSTANCE taxi / number of taxis in city

_____CLASS _____INSTANCE fridge / number of milk cartons in fridge