1. **(4 points) Counting Evaluator**

When we draw environment diagrams, we label the frames $G, E_1, E_2, \ldots$. In the interpreter, we want to have access to the index of the frame that we have created. The way we’re going to do this is have every frame have a variable called **frame-counter** that hold the index of the frame. Notice that every frame that is created has a binding for this variable. There is no binding for **frame-counter** in the global environment.

**Hint:** Every time a compound procedure is invoked, we should increment the counter by one.

**Hint Hint:** It might help to draw the environment diagram for this interaction, taking into account the fact that every frame has a variable called **frame-counter**.

Here’s how the evaluator should behave when you’re done:

```
MCE> (define (foo) **frame-counter**)
MCE> (define (bar) **frame-counter**)
MCE> (define square (lambda (x) (* x x)))
MCE> (foo)
1
MCE> (square 4)
16
MCE> (foo)
3
MCE> (bar)
4
```
2. (4 points) Logic Programming
Write a logic program called sub-filter that filters from a list and keeps only words that contain all letters from the first sentence within it. Assume rules for subset have already been written for you. Examples are shown below. Assume that the letters in the first list are distinct.

Tests for subset

Query Input> (subset (1 3 2) (9 2 5 1 3)) ;this has 1, 2, and 3 in it
Query Output> (subset (1 3 2) (9 2 5 1 3))

Query Input> (subset (1 2) (2 3 4))
Query Output>

Tests for sub-filter

Query Input> (sub-filter (m o n) (monster digimon yugioh pokemon) ?what)
Query Output> (sub-filter (m o n) (monster digimon yugioh pokemon) (monster digimon pokemon))

Query Input> (sub-filter (a i) (reia judy sam chenning alicia vivek yuan) ?who)
Query Output> (sub-filter (a i) (reia judy sam chenning alicia vivek yuan) (reia alicia))

Once again, you do NOT have to write subset, but you should feel free to call within your definition of sub-filter

3. (2 points) Lazy Evaluator
Suppose you type the following into the lazy evaluator:

```
(define (foo x y z)
    x y z)
(define bar 9)
```

What is the value of bar after each of the following interactions?

(a) (1 pt)

`lazy> (foo 3 4 (set! bar (+ bar 3)))`

The value of bar is ____________
(b) (1 pt) Suppose you type the following into the interpreter **instead** of what you typed in part a.

```
lazy> (begin
    (foo (set! bar (+ bar 1)) 4 (set! bar (+ bar 3)))
    bar)
```

The value of `bar` is ___________

4. (3 points) **Counting Palindromes**

Sam loves palindromes. He wants to find out all the palindromes in his book and how many times each palindrome appears. Write a procedure `get-palindromes` and its mapper/reducer that takes in a stream of `kv-pairs` where the key is the number of the line in the book and the value is a line of text from the book. It uses mapreduce to output a new list a `kv-pairs` where the keys are the palindromes in the book and the values are the number of times each palindrome appears.

Assume you have a procedure `palindrome` that takes in a word and returns true if the word is a palindrome, and false otherwise.

```
STk> (get-palindromes '((0 non trivial) (1 non racecar)))
((non 2) (racecar 1))
STk> (get-palindromes '((0 hello world) (1 yep nope)))
()
```