SYSC 3101 Lab 6

Exercise 1

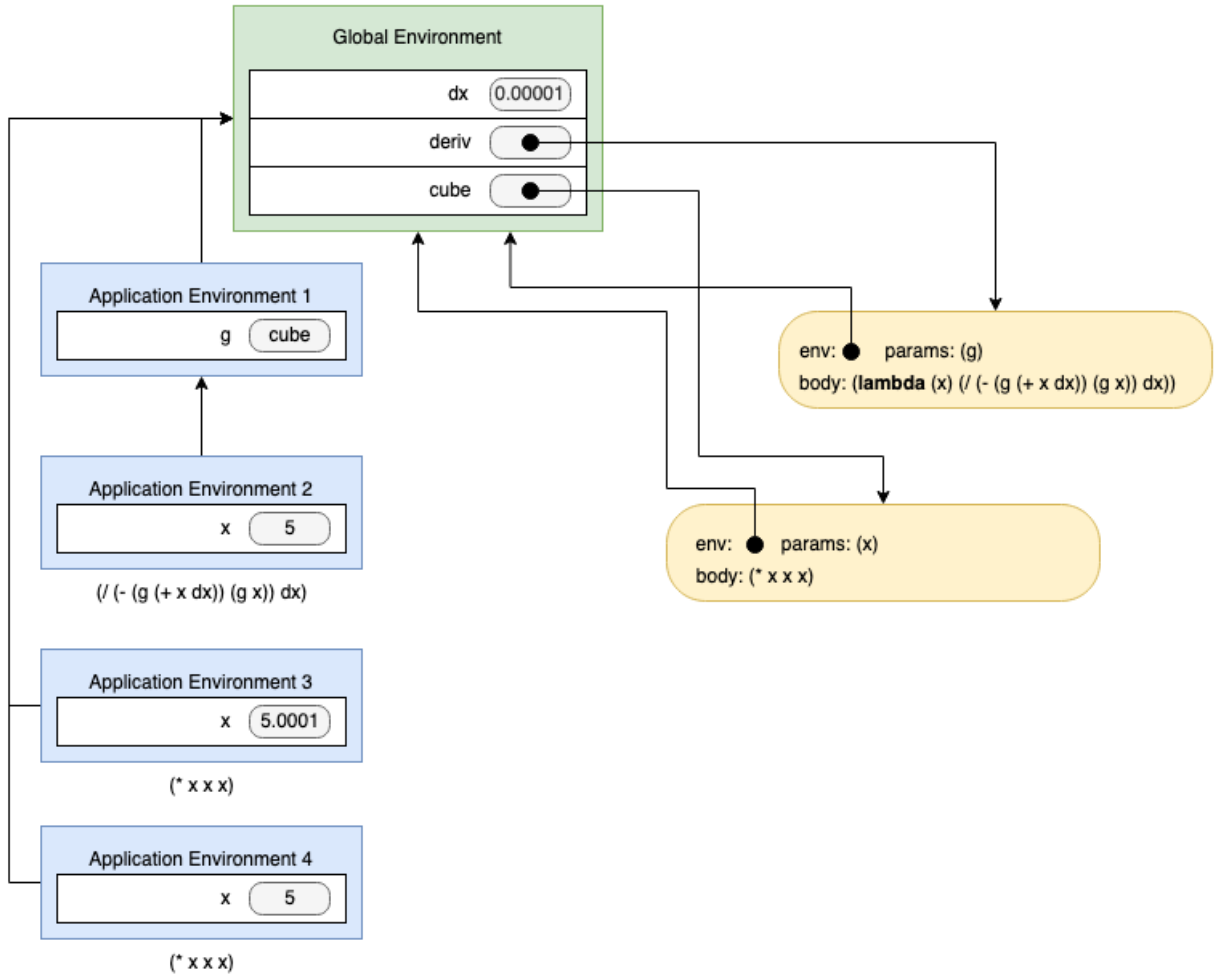
Draw a diagram that depicts the environment created when these expressions are evaluated:

```
(define dx 0.00001)

(define (cube x) (* x x x))

(define (deriv g)
  (lambda (x)
    (/
     (- (g (+ x dx)) (g x))
     dx)
    )
)

((deriv cube) 5)  ; Returns 75.00014999664018
```

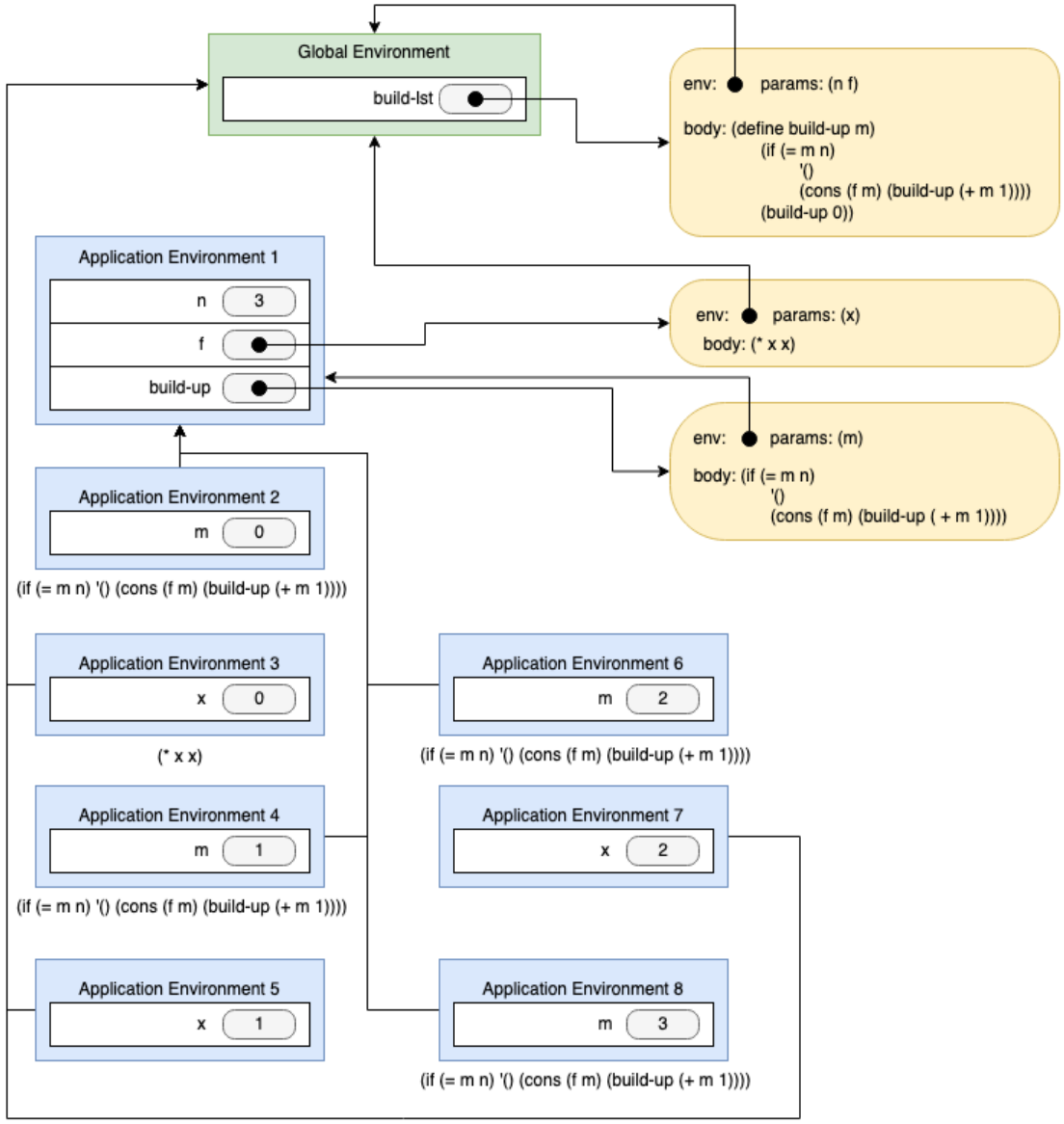


Exercise 2

Draw a diagram that depicts the environment created when these expressions are evaluated:

```
(define (build-list n f)
  ; build-up creates a list by applying procedure f to the integers
  ; from m to n-1, m < n, in order. The first list element is the
  ; value produced by (f m), and the last list element is the value
  ; produced by (f (- n 1)).
  (define (build-up m)
    (if (= m n)
        '()
        (cons (f m) (build-up (+ m 1)))))
  (build-up 0))


(build-list 3 (lambda (x) (* x x)))
```



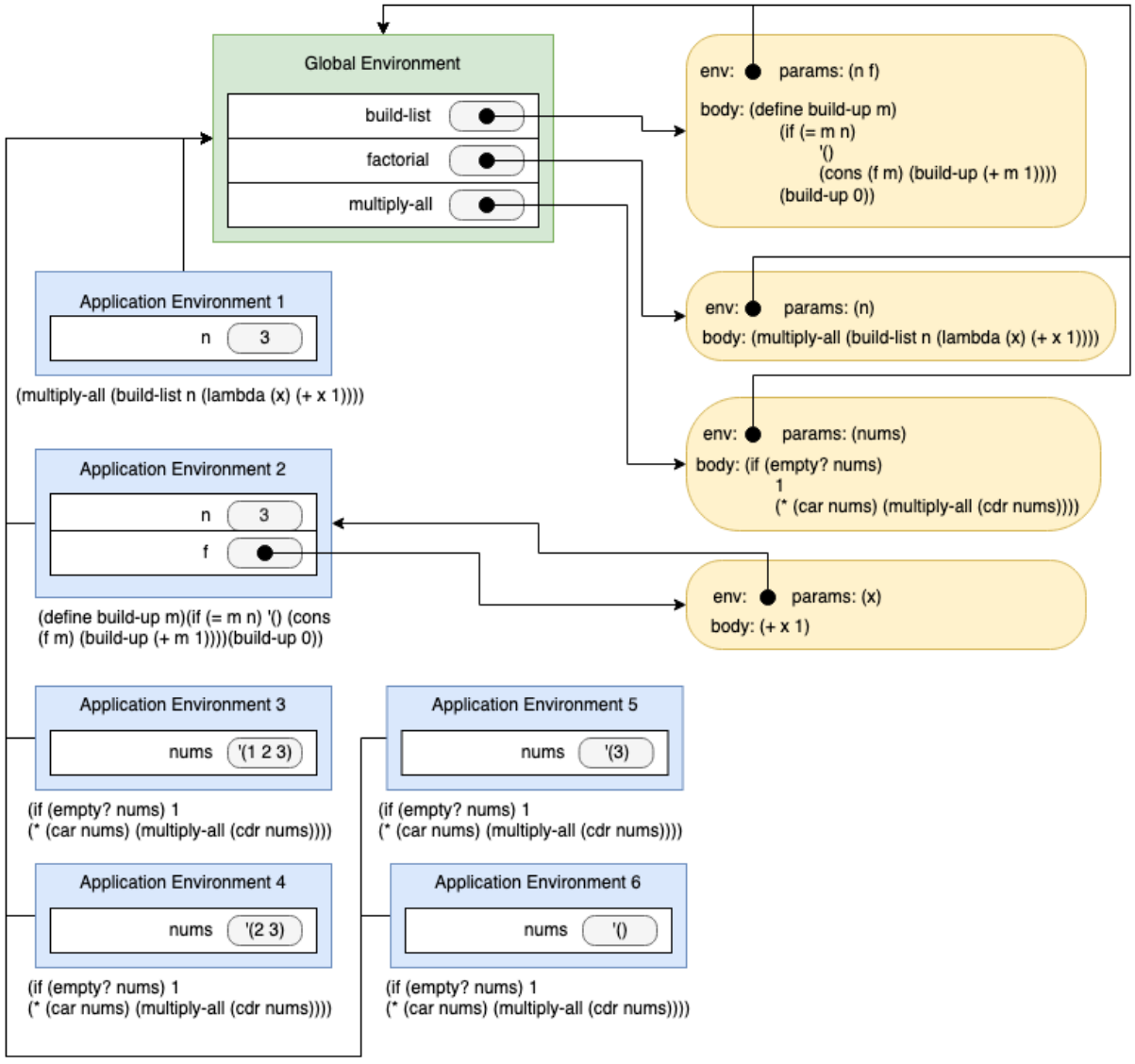
Exercise 3

Draw a diagram that depicts the environment created when these expressions are evaluated:

```
(define (multiply-all nums)
  (if (empty? nums)
      1
      (* (car nums) (multiply-all (cdr nums)))))

(define (factorial n)
  (multiply-all (build-list n (lambda (x) (+ x 1)))))

(factorial 3)
```



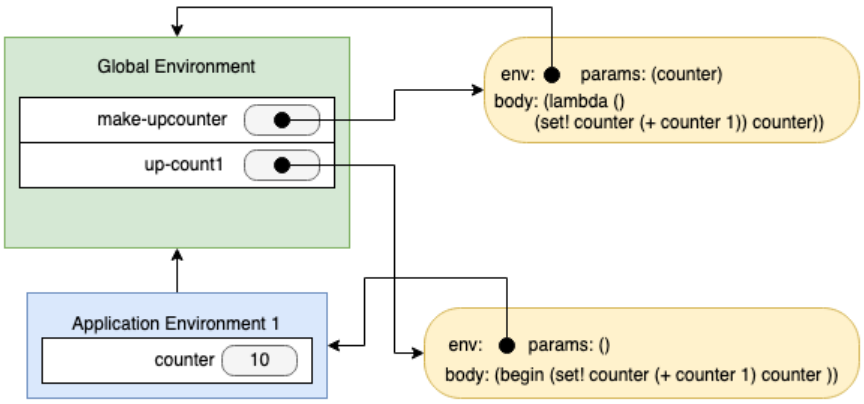
Exercise 4

Recall this example:

```
(define (make-upcounter counter)
  (lambda ()
    (set! counter (+ counter 1))
    counter))
```

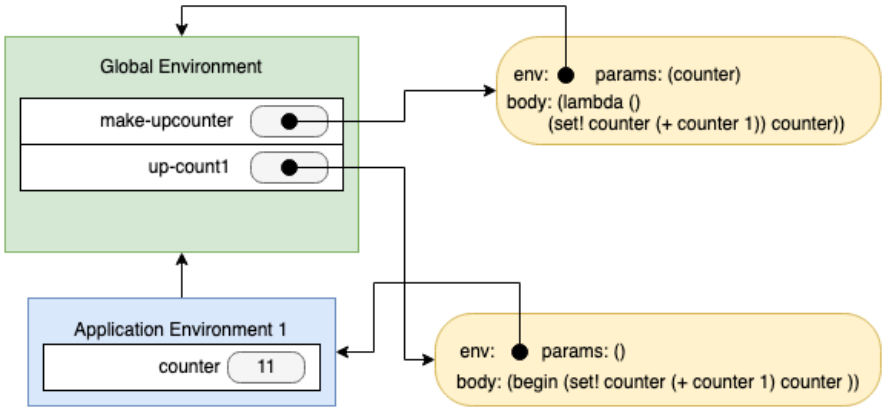
a) Draw a diagram that depicts the environment created by:

```
(define up-count1 (make-upcounter 10))
```



b) Draw a diagram that depicts the environment that is created the first time the following expression is evaluated:

```
(up-count1)
```



c) The lab code has a parameter to set the counter's initial value, whereas in the textbook it does not have a parameter and uses a *let* statement. This impacts the counter in the application environment as the *let* statement binds the counter to 0, whereas the lab code binds it to the parameter, resulting in different counter values.