Environment Model
MIT 6.001 Recitation

High level issues

In practice, how do you create a new frame?

What is the difference between `set!` and `define` ?

How do you access information from a child frame?

Factorial

Evaluate the expressions below, following the rules of the environment model. Draw the corresponding environment diagram.

```
(define (fact n)
  (if (= n 1)
    1
    (* n (fact (- n 1)))))

(define n 6)
(fact 3)
```

Let

```
(let ( (x (+ 2 5))
   (y 7))
  (*x y))
```

Prev

How do you create a procedure that takes an argument and returns the previous argument it was called with?

```
(prev 1)    ➔'undefined
(prev #t)   ➔1
(prev 4)    ➔#t
```

Procedure counter

We want to write a higher-order procedure that turns a procedure into a procedure with a counter. That is, the new procedure still returns the output of the input procedure, but in addition, it increments a counter. When called with the magic symbol ‘count, the new procedure returns the counter. We are going to study three attempts and whether they might succeed or fail.

Version 1

```
(define make-count-proc-1
  (lambda(f)
    (lambda(x)
      (let ((count 0))
        (cond ((eq? x ‘count) count)
          (else
            (set! count (+ count 1))
            (f x)))))))

(define sqrt-c-1 (make-count-proc-1 sqrt))
(sqrt-c-1 4)        ➔ ?
(sqrt-c-1 ‘count)   ➔ ?
```

Version 2

```
(define make-count-proc-2
  (lambda(f)
    (let ((count 0))
      (lambda(x)
        (cond ((eq? x ‘count) count)
          (else
            (set! count (+ count 1))
            (f x)))))))

(define sqrt-c-2 (make-count-proc-2 sqrt))
(define square-c-2 (make-count-proc-2 square))

(sqrt-c-2 4)        ➔ ?
(sqrt-c-2 ‘count)   ➔ ?

(square-c-2 4)      ➔ ?
(square-c-2 ‘count) ➔ ?
```

Version 3

```
(define make-count-proc-3
  (let ((count 0))
    (lambda(f)
      (lambda(x)
        (cond ((eq? x ‘count) count)
          (else
            (set! count (+ count 1))
            (f x)))))))

(define sqrt-c-3 (make-count-proc-3 sqrt))
(define square-c-3 (make-count-proc-3 square))

(sqrt-c-3 4)        ➔ ?
(sqrt-c-3 ‘count)   ➔ ?

(square-c-3 4)      ➔ ?
(square-c-3 ‘count) ➔ ?
```