MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall 2007

Recitation 14 — 10/24/2007
More Environment Diagrams

Three Counter Attempts

1. ```
   (define make-count-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-1 sqrt))

   (sqrt-c-1 4)
   (sqrt-c-1 'count)
   ```
   ```

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

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

(sqrt-c-2 4)
(sqrt-c-2 'count)

(square-c-2 4)
(square-c-2 'count)
```

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

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

(sqrt-c-3 4)
(sqrt-c-3 'count)

(square-c-3 4)
(square-c-3 'count)
```

4. The procedure `last-pair` returns the last pair of a list (guaranteed to have '() in the cdr).

```
(define (list-inserters lst)
  (let ((last (last-pair lst)))
    (list (lambda (x)
            (set-cdr! lst (cons x (cdr lst)))
            lst)
          (lambda (y)
            (set-cdr! last (cons y '()))
            (set! last (cdr last))
            lst))))

(define the-list (list 1 3 4))

(let ((ins (list-inserters the-list)))
  ((car ins) 2)
  ((cadr ins) 5))
```

Finish the environment diagram.