Scoping, `define` versus `set!`, and Shadowing

```
(define x 0)
(define f
  (lambda (y)
    (define x (+ y 10))
    x))
(define g
  (lambda (y)
    (set! x (+ y 10))
    x))
```

Find the values of:

(f 5) \[15\], \(x\) \(0\), (g 5) \[15\], and \(x\) \(5\)

...and show the environment diagram:

![Environment Diagram]

Nameless Wonders

```
(define x 3)
((lambda (x y) (+ (x 1) y))
  (lambda (z) (+ x 2))
  3)
```

What is the value of this expression?  \[8\]

*Show all relevant portions of the environment diagram used to evaluate this block of code.*

![Environment Diagram for Nameless Wonders]
Aspartame (Desugaring let)

Desugar the following expression:

```
(define x 4)
(let ((x (+ 2 1))
     (y (square x)))
  (* x y))
```

; DESUGARS TO:

```
(((lambda (x y) (* x y))
  (+ 2 1))
 (square x)) ;==> 48
```

Show all relevant portions of the environment diagram used to evaluate this block of code.

\[\lambda\text{-let}\]

```
(define x 5)
(let ((x (lambda (x) (+ 6 x)))
     (set! x (x 7))
     x)
```

What is the value of this expression? 13

Show all relevant portions of the environment diagram used to evaluate this block of code.
Yet More Complexity

(define a 5)
(define foo
  (let ((a 10))
    (lambda (x)
      (+ x a))))
(define (bar a) (foo 20))
(bar 100)

What is the value of this expression? 30

Show all relevant portions of the environment diagram used to evaluate this block of code.
Insanity!

(define (make-count-proc f)
  (let ((count 0))
    (lambda (x)
      (if (eq? x 'count)
          count
          (begin (set! count (+ count 1))
                 (f x))))))

(define sqrt* (make-count-proc sqrt))
(define square* (make-count-proc square))

Find the values of:
(sqrt* 4) 2,
(sqrt* 'count) 1,
(square* 4) 16,
and (square* 'count) 1

...and show the environment diagram: