LAMBDA, LET, AND ENVIRONMENTS.

Draw the environment diagrams that would result from the following code sequences being evaluated. Assume that the evaluation occurs in the context of the global environment.

Draw a frame for the global environment and any other frames that are created. Indicate symbol bindings that occur in any frame, arrows that lead up from sub-environments to higher ones, and procedure objects (using the double-circle representation of procedures).

If a frame is created and then gets garbage-collected by the end, indicate that by drawing the frame with a dashed line.

Each problem with the same number is a continuation of the previous problem. Copy anything you created in the previous subproblem to the new one and add to it.

New problem numbers start a new problem.

Problem 1a.

(define (square n)
  (* n n))
Problem 1b.

(define foo 3)

Problem 1c.

(define bar (square foo))
Problem 2a.

(define (proc x y)
  (let ((a (+ x y))
        (b (- x y)))
    (* a b)))

Problem 2b.

(proc 3 4)
Given the environment diagram, write Scheme code that could have been evaluated to create it.

**Problem 3a.**

```
(make-expt
  (lambda (n)
    (expt n n)))
```

**Problem 3b.**

```
(make-expt
  (lambda (n)
    (expt n n))
  2)
```