Overview

This problem set deals with state, mutation, and the environment model. Much of the problem set involves drawing box-and-pointer diagrams and environment diagrams. These parts are easiest to do with pencil and paper rather than typing them in.

Remember to draw arrows from the variables bound to box and pointer diagrams to the diagrams. Remember to name your environments in environment diagrams so that you can say which expression is evaluated relative to which environment.

Any code should be written using the MzScheme language in DrScheme, so that error is defined. Turn in your code as ps7-ans.ss

Problems

Problem 1  Do Exercise 3.1 on p. 224. Also draw an environment diagram for the code in the problem.

Problem 2  Do Exercise 3.3 on p. 225.

Problem 3  This problem compares the environment model and the substitution model on code where both should give the same answer. You need to know the rules for variables, combinations, primitive procedures and compound procedures in both models.

Write out the steps needed to evaluate the following code with
(a) the substitution model
(b) the environment model. (Draw the environment diagram and label the various environments. For each expression say what environment you are evaluating it in.)

```
(define (compose f)
  (lambda (g)
    (lambda (x) (f (g x)))))
(define (square x) (* x x))
(define (add1 y) (+ y 1))
(((compose add1) square) 5)
```
Problem 4  Do Exercise 3.8 on p. 236.

Problem 5  Do Exercise 3.12 on p. 255.

Problem 6  Do Exercise 3.15 on p. 259.

Problem 7  Do Exercise 3.16 on p. 259.

Problem 8  Do Exercise 3.22 on p. 266.
The pattern of the code is similar to the make-account code on page 223, or the rewrite of cons on the bottom of page 260 and top of page 261. In the bank code, the mutable value balance is passed as part of make-account. In make-queue, the actual structure of queues is hidden from the callers, so the mutable parameters are initialized by a let expression. By convention we use a let expression rather than a define in a situation like this; define is used for defining constants (for instance pi*2) or procedures.