

## LAB ASSIGNMENT 7B

1. Lazy evaluator. Abelson and Sussman, exercises 4.27 and 4.29.
2. In this exercise we learn what a *continuation* is. Suppose we have the following definition:

```
(define (square x cont)
  (cont (* x x)))
```

Here `x` is the number we want to square, and `cont` is the procedure to which we want to pass the result. Now try these experiments:

```
> (square 5 (lambda (x) x))
> (square 5 (lambda (x) (+ x 2)))
> (square 5 (lambda (x) (square x (lambda (x) x))))
> (square 5 display)
> (define foo 3)
> (square 5 (lambda (x) (set! foo x)))
> foo
```

Don't just type them in – make sure you understand why they work! The nondeterministic evaluator works by evaluating every expression with *two* continuations, one used if the computation succeeds, and one used if it fails.

```
(define (reciprocal x yes no)
  (if (= x 0)
      (no x)
      (yes (/ 1 x))))

> (reciprocal 3 (lambda (x) x) (lambda (x) (se x '(cannot reciprocate))))
> (reciprocal 0 (lambda (x) x) (lambda (x) (se x '(cannot reciprocate))))
```