**CS 61A Week 8**

Topic: Assignment, state, environments

**Midterm Wednesday , 7–9pm.**

**Reading:** Abelson & Sussman, Section 3.1, 3.2

Also read "Object-Oriented Programming—Below-the-line view" (in course reader).

**Homework:**

Abelson & Sussman, exercises 3.3, 3.4, 3.7, 3.8, 3.10, 3.11

**Note:** Part I of programming project 3 is also due next week.

**Extra for experts:**

The purpose of the environment model is to represent the scope of variables; when you see an `x` in a program, which variable `x` does it mean?

Another way to solve this problem would be to *rename* all the local variables so that there are never two variables with the same name. Write a procedure `unique-rename` that takes a (quoted) lambda expression as its argument, and returns an equivalent lambda expression with the variables renamed to be unique:

```
> (unique-rename '(lambda (x) (lambda (y) (x (lambda (x) (y x))))))
(lambda (g1) (lambda (g2) (g1 (lambda (g3) (g2 g3)))))
```

Note that the original expression had two variables named `x`, and in the returned expression it's clear from the names which is which. You'll need a modified counter object to generate the unique names.

You may assume that there are no `quote`, `let`, or `define` expressions, so that every symbol is a variable reference, and variables are created only by `lambda`.

Describe how you'd use `unique-rename` to allow the evaluation of Scheme programs with only a single (global) frame.

---

Unix feature of the week: `foreach`, `grep`, `find`

Emacs feature of the week: `C-t` (transpose), `M-c`, `M-u`, `M-l` (change case)