Solutions to In-Class Problems — Week 5, Mon

The Variable Convention

Problem. A Scheme expression satisfies the "Variable Convention" if no variable identifier is bound more than once, and no identifier has both bound and unbound occurrences. For example, the expression

```
(let ((x 2) (y 5))
  (+ ((lambda (x) (+ x 1)) 3) ((lambda (z) (+ x y z 11)) 99) z)).
```

violates the Variable Convention because `x` is bound twice—once by `let` and once by `lambda`, and also because `z` has both a bound and an unbound occurrence.

Any expression can be slightly modified to satisfy the Convention solely by adding integer suffixes to some of the bound identifiers—in a way that preserves all the binding structure and all the computational behavior of the original expression.

For example, by adding suffix `0` to the `x`'s and `z`'s bound by the `lambda`'s, we obtain an equivalent expression which satisfies the Variable Convention:

```
(let ((x 2) (y 5))
  (+ ((lambda (x0) (+ x0 1)) 3) ((lambda (z0) (+ x y z0 11)) 99) z)).
```

Show how to add such suffixes to the identifiers in

```
(a b c d e
   (let ((a e) (b c))
     (a b c d e
        (letrec ((a c)(c b))
          (a b c d e)))))
```

to obtain an equivalent expression satisfying the Variable Convention. (See the Scheme reference manual to find out the scoping rules for `letrec`.)

SOLUTION:

```
(a b c d e
   (let ((a0 e) (b0 c))
     (a0 b0 c d e
        (letrec ((a1 c0)(c0 b0))
          (a1 b0 c0 d e)))))
```