MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall 2007

Recitation 13 — 10/19/2007
Environment Model

Eval

- *name* - Look up *name* in the current environment, if found return value, otherwise lookup in enclosing (parent) environment frames.
- `(lambda (`*params*`)` *body*`)` - Create double bubble with code ptr to *params* and *body* and env ptr to current environment.
- `(define` *name value*`)` - Evaluate *value* and then create/replace binding for *name* with the result.
- `(set!` *name value*`)` - Evaluate *value* and then replace the first binding for *name* in the chain of environments, starting with the current env.
- `(`*proc args ...* `)` - Evaluate *proc* and *args* in the current environment. If *proc* results in a compound procedure, then apply, otherwise just compute the result.
- Otherwise – Follow the correct rule (numbers, if, cond, begin, quote, etc.)

Apply

- Step 1 - Drop a new frame
- Step 2 - Link frame pointer of new frame to environment pointed to by env pointer of double bubble being applied.
- Step 3 - Bind params of double bubble in the new frame.
- Step 4 - Eval the *body* in the new frame.

Problems

Problem 1

```
(define (square x)
  (* x x))
(define (sum-of-squares x y)
  (+ (square x) (square y)))
(sum-of-squares 2 3)
```

Problem 2

```
(define x 3)
((lambda (x y) (+ (x 1) y))
 (lambda (z) (+ x 2))
 3)
```

Problem 3

```
(define (fact n)
  (if (= n 0)
      1
      (* n (fact (- n 1)))))
(fact 2)
```

Problem 4

```
(define x 1)
(let ((x 5)
      (y (+ x 5)))
  (+ x y))
```

Problem 5

```
(define (previous f)
  (let ((old #f))
    (lambda (x)
      (let ((return old))
        (set! old (f x))
        return))))
(define echo (previous (lambda (y) y)))
(echo 1)
```