MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 1999

Recitation – Wednesday, February 3

0. Announcements

- **Section Staff:**

RI:	Mike Leventon	`leventon@ai.mit.edu`	
TA:	Sandia Ren	`sren@mit.edu`	
TA:	Hooman Vassef	`hvassef@mit.edu`	(Half of section 12)

- **Collaboration Policy:** Read carefully in the handout.
- **Attendance:** Lecture recommended. Section and Tutorials required.
- **Problem Sets:** No late problem sets accepted. Psets are an important part of your grade. Start them early!
- **Course Web Page:** `http://mit.edu/6.001`
- **Section Web Page:** `http://www.ai.mit.edu/people/leventon/6001`
  Section notes (with solutions) will be posted here.

1. Rules for Scheme

To evaluate an expression, follow these rules:

- A **numeral** evaluates to the **number**
- A **name** evaluates to **the value associated with that name**
- A **lambda expression** evaluates to a **procedure object**
- A **combination** is evaulated as follows:
  1. **Evalute** the *subexpressions in any order*
  2. **Apply** the *value of the operator subexpression* to the *values of the remaining subexpressions.*

To apply a procedure to its arguments, evaluate the body of the procedure where each parameter is replaced by the corresponding value.

To evaluate a define, evaluate the body of the define, and associate that value with the name listed in the define.

2. Simple Examples

To what do the following expressions evaluate (assume they are evaluated in sequence)?

```
⇒ 7

⇒ -

⇒ (+ 2 4)

⇒ (* (- 5 3) (/ 9 3))

⇒ (7 - 4)
```

3. More Examples

To what do the following expressions evaluate (assume they are evaluated in sequence)?

```
⇒ (lambda (x) (* x x))

⇒ ((lambda (x) (* x x)) 5)

⇒ (define double (lambda (x) (* 2 x)))

⇒ (double (double 6))

⇒ (double double)
```

```
⇒ (define cube (lambda (x) (*x x x)))

⇒ (cube 3)

⇒ (define + 3)

⇒ (define - 6)

⇒ (* + -)
```

4. Writing a Procedure

Define a procedure called `average` that computes the average of its two numeric arguments.

5. Substitution Model

Use the substitution model to evaluate the following expression: `(average 4 (double 4))`

6. Subtleties

Consider the following two definitions below. How are they similar and how do they differ?

```
(define plus +)

(define add
  (lambda (x y)
    (+ x y)))
```

7. More Subtleties

What do think should be the values of the following expressions? (Note: these are **not** things you have to memorize.)

```
⇒ (+ 4)

⇒ (- 3)

⇒ (/ 5)

⇒ (/ 60 5 2 3)
```

```
⇒ (+)

⇒ (*)

⇒ (-)
```