

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
Department of Electrical Engineering and Computer Science  
6.090—Building Programming Experience  
IAP 2005

**Lecture 3**

## Scheme

### 1. Special Forms

(a) *begin* - (`begin` *exps*)

Evaluate each expression in order and return the value of the last expression.

## Problems

1. Write `abs`, a function that returns the absolute value of its input.

```
(abs 5)
;Value: 5
(abs -5)
;Value: 5
```

```
(define abs
  (lambda (val)
```

2. Write `sum-to-n` which sums the numbers from 1 to `n` inclusive.

```
(define sum-to-n
  (lambda (n)
```

Alter the procedure to sum the squares of the numbers.



## Tower of Hanoi

### Scheme

#### 1. Special Forms

- (a) *define* (sugared form) - (`define` (*name parameters*) *expressions*)  
This form is equivalent to (`define` *name* (`lambda` (*parameters*) *expressions*)).

- (b) *let* - (`let` *bindings* *body*)  
Binds the given bindings for the duration of the body. The bindings is a list of (*name value*) pairs. The body consists of one or more expressions which are evaluated in order and the value of last is returned.

### Problems

7. Guess the value, then evaluate the expression in scheme. If your guess differs from the actual output, try desugaring any relevant expressions.

```
(define (foo x)
  (+ x 3))
```

foo

```
(foo 5)
```

```
(define bar 5)
```

```
(define (baz) 5)

bar

baz

(bar)

(baz)

(let ((a 3)
      (b 5))
  (+ a b))

(let ((+ *)
      (* +))
  (+ 3 (* 4 5)))

(define m 3)
(let ((m (+ m 1)))
  (+ m 1))

(define n 4)
(let ((n 12)
      (o (+ n 2)))
  (* n o))
```

## Guessing Game