

# CS-257L

## Nonimperative Programming: Scheme!

Instructor:

Joel Castellanos

e-mail: [joel@unm.edu](mailto:joel@unm.edu)

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering  
Center (FEC) room 321

```
(define prime-factors (lambda (n)
  (if (< n 1)
      "Bad Input: Expected a positive
      whole number."
      (if (not (= n (truncate n)))
          "Bad Input: Expected a positive
          whole number."
          (try n 2 ())))))
```

# Homework – Due Wednesday – Feb. 13

---

- Read Chapter 5 "*Oh My Gawd: It's Full of Stars*" of *The Little Schemer*
- Nothing to hand-in.
- Perhaps a quiz.

# Problem:

## Formatted with `Display` is Verbose

---

How can the text and values be printed more compactly?

```
(define b 3)
(define h 7)
(begin
  (display "The base = ")
  (display b)
  (display " cm, the height = ")
  (display h)
  (display " cm")
  (newline)
)
```

The base = 3 cm, the height = 7 cm

# Quasiquote "`" and unquote ", "

---

Syntax: ``<qq template>`

"Backquote" or "quasiquote" expressions are useful for constructing a list structure when most but not all of the desired structure is known in advance.

If no commas appear within the `<qq template>`, the result of evaluating ``<qq template>` is equivalent to the result of evaluating `'<qq template>`.

If a comma appears within the `<qq template>`, however, the expression following the comma is evaluated ("unquoted") and its result is inserted into the structure instead of the comma and the expression.

# Quasiquote Example

```
(begin
  (display "The base = ")
  (display b)
  (display " cm, the height = ")
  (display h)
  (display " cm")
  (newline)
)
(begin
  (display `("The base =" ,b "cm, the height =" ,h "cm"))
  (newline)
)
```

The base = 3 cm, the height = 7 cm

(The base = 3 cm, the height = 7 cm)

# Display with multiple arguments + newline

---

```
(define display2
  (lambda (displaylist)
    (if (null? displaylist) (display "")
        (begin
          (display (car displaylist))
          (display2(cdr displaylist))
        )
    )
  )
)
(begin
  (display2
    `("The base = " ,b " cm, the height = " ,h " cm"))
  (newline)
)
```

# The Little Schemer: Chapter 3: rember

---

- |                               |                |
|-------------------------------|----------------|
| 1. (rember 'c '(a b c d e))   | 1. (a b d e)   |
| 2. (rember 'b '(a b a b a b)) | 2. (a a b a b) |
| 3. (rember 'a '(x y z))       | 3. (x y z)     |
| 4. (rember 'a ())             | 4. ()          |
| 5. (rember 'a 'a)             | 5. error       |

# How does this work?

```
(define rember
  (lambda (a lat)
    (cond
      ((null? lat) (quote ()))
      (else (cond
        ((eq? (car lat) a) (cdr lat))
        (else (cons (car lat)
                    (rember a (cdr lat))))
      )
    )
  )
)
(rember 5 (1 2 5 6))
```

# Scheme and Abstract Algebra

```
(define o+  
  (lambda (n m)  
    (cond  
      ((zero? m) n)  
      (else (add1 (o+ n (sub1 m)))))  
    )  
  ))
```

(o+ 3 0) → 3

(o+ 3 2) ↓ (o+ 3 1) ↓ (o+ 3 0) ↓ 3 ↑  
(add1 3) ↑ (add1 4) ↑ 5