# Tutorial 4: Scintacks and Some Antics (Syntax and Semantics)

CS 135 Fall 2007

October 3-5, 2007

This week we will be looking at the mathematical *structure* and *meaning* of a Scheme program. The material here comes primarily from Lecture Module 4 in the class notes. It's quite fortunate (and unusual) that we are working in a programming language in which we can rigorously define the meaning of every program with relatively few rules. The only unfortuanate side effect is that we'll have to endure a week without the fun stories and problem descriptions that we've all grown accustomed to in these tutorials.

Give a full syntactic/semantic analysis of each of the following Scheme programs. That is, go through (one by one) each of the substitution steps to completely evaluate each expression to a value. If an error occurs, making evaluation impossible, pinpoint the exact location and nature of the error (syntax, semantics, or other).

1.
```
(and (symbol? 'hello)
     (= (- 5 1) (* 2 3))
     (/ "a string" "another string"))
```

2.
```
(define a (+ 2 3))
(define (foo2 x)
  (cond [(or (> x 1)
     (< x -1))
 (sqr x)]
[(zero? x) 1]))
(foo2 a)
(foo2 (/ a a))
```

3.
```
(define (foo3 5)
  (+ 1 5))
(/ (foo3 5)
   0)
```

4.
```
(define-struct name (first middle last))
(define (foo4 nme)
  (name-middle (+ nme 1)))
(name-last (make-name "James" "A" "Garfield"))
```

5.
```
(define (foo5 x)
  (cond [(= 1 x) 2]
[else
  (* 2
     (foo5 (sub1 x)))]))
(foo5 3)
(foo5 -2)
```