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

Lecture 6

Scheme

1. **Special Forms**

   (a) *and* - `(and` *arg1 arg2 ...*`)`
   Evaluates arguments from left to right, stopping at the first one that evaluates to false and returning false. Should all the arguments evaluate true-ishly, returns the value of the last argument.

   (b) *or* - `(or` *arg1 arg2 ...*`)`
   Evaluates arguments from left to right, stopping at the first one that evaluates to true-ish and returns that value. Should all the arguments evaluate to false, returns false.

Higher Order Procedures

```
(define sum
  (lambda (f x y dx)
```

Types

Problems

For each expression, write the type of the **value** that results from evaluating the expression. Ignore `define` expressions.

```
4

(+ 1 1)

(lambda (x) (+ x 1))

(lambda (x) (= x 1))

(define square
  (lambda (x) (* x x)))

square

(square 5)

(define a
  (lambda (f) (+ (f 5) 1)))

a

(a square)

(define b
  (lambda (x y)
    (+ (a x) y)))

b

(b square 4)

(define c
  (lambda (x)
    (lambda (y)
      (+ x y))))

c

(c 5)
```