

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring 2004

Recitation 2
More Scheme

Scheme

1. Special Forms

(a) *if* - (`if test consequent alternative`)

If the value of the test is not false (`#f`), evaluate the consequent, otherwise evaluate the alternative.

(b) *lambda* - (`lambda parameters body`)

Creates a procedure with the given parameters and body. Parameters is a list of names of variables. Body is one or more scheme expressions. When the procedure is applied, the body expressions are evaluated in order and the value of the last one is returned.

Problems

1. Evaluate the following expressions (assuming `x` is bound to 3):

```
(if #t (+ 1 1) 17)
```

```
(if #f #f 42)
```

```
(if (> x 0) x (- x))
```

```
(if 0 1 2)
```

```
(if x 7 (7))
```

2. Evaluate the following expressions:

(lambda (x) x)

((lambda (x) x) 17)

((lambda (x y) x) 42 17)

((lambda (x y) y) (/ 1 0) 3)

((lambda (x y) (x y 3)) (lambda (a b) (+ a b)) 14)

3. Suppose we're designing an point-of-sale and order-tracking system for Wendy's¹. Luckily the Über-Qwüick drive through supports only 4 options: Classic Single Combo (hamburger with one patty), Classic Double With Cheese Combo (2 patties), and Classic Triple with Cheese Combo (3 patties), Avant-Garde Quadruple with Guacamole Combo (4 patties). We shall encode these combos as 1, 2, 3, and 4 respectively. Each meal can be *biggie-sized* to acquire a larger box of fries and drink. A *biggie-sized* combo is represented by 5, 6, 7, and 8 respectively.
- Write a procedure named `biggie-size` which when given a regular combo returns a *biggie-sized* version.
 - Write a procedure named `unbiggie-size` which when given a *biggie-sized* combo returns a non-*biggie-sized* version.
 - Write a procedure named `biggie-size?` which when given a combo, returns true if the combo has been *biggie-sized* and false otherwise.
 - Write a procedure named `combo-price` which takes a combo and returns the price of the combo. Each patty costs \$1.17, and a *biggie-sized* version costs \$.50 extra overall.

¹6.001 and MIT do not endorse and are not affiliated with Wendy's in any way. They merely capitalize on the pleasant way "biggie-size" rolls off the tongue.

- (e) An order is a collection of combos. We'll encode an order as each digit representing a combo. For example, the order 237 represents a Double, Triple, and *biggie-sized* Triple. Write a procedure named `empty-order` which takes no arguments and returns an empty order.
- (f) Write a procedure named `add-to-order` which takes an order and a combo and returns a new order which contains the contents of the old order and the new combo. For example, `(add-to-order 1 2) -> 12`.
- (g) Write a procedure named `order-size` which takes an order and returns the number of combos in the order. For example, `(order-size 237) -> 3`.
- (h) Write a procedure named `order-cost` which takes an order and returns the total cost of all the combos. You may find `quotient` (integer division) useful.