COMP 210, Spring 2001
Lecture 6: Even More Tricks with Lists

Reminders:

- First Exam: February 23 in class

Review

1. Did more work with lists, introduced Scheme's built-in list construct, based on **`cons`**, **`first`**, **`rest`**, and **`cons?`**
2. Talked some about data definitions and when we need them. At this point in COMP 210, we want you to write a data definition for each list construct (even though you use **`cons`** *et al.*) because the data definition specifies what kind of object is going into the list (list-of-symbol versus list-of-natnum versus list-of-plane).
3. Talked some about templates. The template relates directly to a data-definition. We write a separate template for each kind of information that needs a data definition.

Back to JetSet Airlines

At the end of class, I asked you to solve the following problem. Write a program that consumes a list-of-planes and produces a list containing all the planes that are DC-10s. This is my version.

```
(define (just-dc10s a-lop)
  (cond
    [(empty? a-lop) empty]
    [(cons? a-lop)
     (cond [(isdc10? (first a-lop))
              (cons (first a-lop) (just-dc10s (rest a-lop)))]
            [else (just-dc10s (rest a-lop))])]))


;; isdc10? Plane -> boolean
;; Purpose: returns true if p is a DC10, false otherwise
(define (isdc10? p) (symbol=? (Brand-type (Plane-brand p)) 'DC-10))
;; count-dc10s: list-of-plane -> number
;; Purpose:  consumes a list-of-plane and returns the number that
;;           are DC-10s
(define (count-dc10s a-lop)
  (cond
    [(empty? a-lop)  0]
    [(cons?  a-lop)
     (cond [(isdc10? (first lop)) (add1 (count-dc10s (rest a-lop)))]
            [else  (count-DC-10s (rest a-lop))])]))
```

An alternative way to write this program is:

```
;; just-DC-10s: list-of-plane -> number
;; Purpose:  consumes a list-of-plane and returns the number that
;;           are DC-10s
(define (count-DC-10s a-lop)
  (cond
    [(empty? a-lop) 0]
    [(cons? a-lop)
      (add
        (cond
          [(isdc10? (first a-lop)) 1]
          [else 0])
        (count-DC-10s (rest a-lop)))]))
```

Is this acceptable? This brings us back to the heart of COMP 210. COMP 210 is a course about a data-driven, design methodology for programming in the small. This program is consistent with our design recipe but it is not as easy to understand as our original solution. Embedding conditionals in arithmetic expressions should be avoided unless it significantly shortens the code. But, in such a case, it is still better to encapsulate the conditional as a separate "help" function.

A More Complex Variation

Write a program **`all-the-brand`** that consumes a list-of-plane and a symbol and produces a list-of-plane containing all the planes whose brand matches the symbol. Build on your knowledge from **`just-dc10s`**. You can use the same data-definitions and example data.

```
;; all-the-brand : list-of-plane symbol -> list-of-plane
;; Purpose: consumes a list-of-plane and produces a list-of-plane
;;          that contains all the planes whose Brand-type matches
;;          the second argument
(define (all-the-brand a-lop kind) … )

;; Templates
;;
;; for plane
;; (define (is-plane-type a-plane a-symbol)
;;    ( ... (Plane-tailnum  a-plane) ...
;;      ... (plane-brand   a-plane) ...
;;      ... (plane-miles     a-plane) ...
;;      ... (plane-mechanics a-plane) ... ))
;;
;;
;;for list-of-planes
;;(define (all-the-brand a-lop a-symbol)
;;  (cond
;;    [(empty? a-lop) ... ]
;;    [(cons?  a-lop) ... (first a-lop) …
;;                    (all-the-brand (rest a-lop) a-symbol)]))

;; all-the-brand : list-of-plane symbol -> list-of-plane
;; Purpose: consumes a list-of-plane and produces a list-of-plane
;;          that contains all the planes whose type matches the
;;          second argument
(define (all-the-brand a-lop kind)
  (cond
    [(empty? a-lop)    empty]
    [(cons?  a-lop)
       (cond
         [(symbol=? (brand-type (plane-kind (first a-lop))) kind)
          (cons (first a-lop) (all-the-brand (rest a-lop) kind)]
         [else (all-the-brand (rest a-lop) kind)])]))
```