Generic Recursive Procedure

(define *name*
  (lambda (*formal*$_1$ ... *formal*$_n$)
    (cond
      (*base-test*$_1$ *base-consequent*$_1$)
      ...
      (*base-test*$_j$ *base-consequent*$_j$)
      (*recursion-test*$_{j+1}$ *recursive-consequent*$_{j+1}$)
      ...
      (*recursion-test*$_{m-1}$ *recursive-consequent*$_{m-1}$)
      (else *recursive-alternate*$_m$))))

Generic Recursive Procedure

```
(define name
  (lambda (formal_1 ... formal_n)
    (if
      base-test
      base-consequent
      recursive-alternate)))
```

Asymptotic Runtime

- Given that there're no loops, most of the work of determining the big-O characteristics of a procedure lies in determining the number of procedure applications executed by the procedure

Asymptotic Runtime (cont.)

- Simple recursive procedures just recurse on the *cdr* of some list formal
- That makes the number of applications (and, hence, the asymptotic runtime of the procedure)

O(*length of the list*)

<u>as long as the procedure doesn't call any non-O(1) auxiliary procedures</u>

Asymptotic Runtime (cont.)

- If the procedure recurses a linear number of times, and it performs a linear amount of work at each step, that tends to make the number of procedure application

  $$O((\textit{length of the list})^2)$$

  <u>although larger or more complicated runtime orders are certainly possible, as we'll see</u>