```
(require spd/tags)

(@htdd ListOfString)
;; ListOfString is one of:
;;  - empty
;;  - (cons String ListOfString)
;; interp. a list of strings
(define LOS1 empty)
(define LOS2 (cons "Canucks" empty))
(define LOS3 (cons "Leafs" (cons "Canucks" empty)))
(define LOS4 (cons "Canadiens" (cons "Leafs" (cons "Canucks" empty))))

(@dd-template-rules one-of             ;2 cases
                    atomic-distinct    ;empty
                    compound           ;(cons String ListOfString)
                    self-ref)          ;(rest los) is ListOfString

(define (fn-for-los los)
  (cond [(empty? los) (...)]
        [else
         (... (first los)
              (fn-for-los (rest los)))]))

#|
PROBLEM:

Design a function that determines whether "Canucks" appears in a
list of strings.
|#
```

```
(require spd/tags)

(@htdd ListOfNumber)
;; ListOfNumber is one of:
;;  - empty
;;  - (cons Number ListOfNumber)
;; interp. a list of numbers
(define LON1 empty)
(define LON2 (cons 1 (cons 2 (cons 3 empty))))

(@dd-template-rules one-of             ;2 cases
                    atomic-distinct    ;empty
                    compound           ;(cons Number ListOfNumber)
                    self-ref)          ;(rest lon) is ListOfNumber

(define (fn-for-lon lon)
  (cond [(empty? lon) (...)]
        [else
         (... (first lon)
              (fn-for-lon (rest lon)))]))


#|
PROBLEMs:

Design a function that computes the sum of a list of numbers.

Design a function that counts the number of elements in a
list of numbers.

Design a function that produces a new list, where each element
is 2 times the corresponding element in the original list.
|#
```