(defun length1 (list)
    (cond ((null list) 0)
          (t (+ 1 (length1 (cdr list)))))
)

(lisp)

(load "length1.l")

(length1 ()) 0
(length1 nil) 0
(length1 '(a b c)) 3
(length1 '((a b) c (d e f))) 3

(runner%)

(runner%)

(lisp)

(load "myfind.l")

(myfind 'is '(now is the time)) FOUND
(myfind 'for '(now is the time)) NOT-FOUND
(myfind '(a) '(a b c)) NOT-FOUND
(myfind '(b) '(a (b) c)) FOUND
(myfind '(c (d e)) '((a b) (c (d e)) e (f))) FOUND

(runner%)

(runner%)

(lisp)

(load "add.l")

(add '(2)) 2
(add '(2 3)) 5
(add '(2 3 4)) 9
(add nil) 0
(add 3)
>> Error: The value of X, 3, should be a LIST
Back to Lisp Top Level
(add '(2 (3 4)))
>> Error: The value of NUMBER1, (3 4), should be a NUMBER

(runner%)

(runner%)

(lisp)

(load "add-all.l")

(add-all '(1)) 1
(add-all '(2 3)) 5
> (add-all '(3 4 5))
12
> (add-all '(2 (3 4)))
9
> (add-all '((2 3) 4) (1 5)))
15

runner% cat list-atoms.l
(defun list-atoms (list)
  (cond ((null list) nil)
        ((atom (car list)) (list-atoms (cdr list)))
        ((null (car list)) (list-atoms (cdr list)))
        (t (append (list-atoms (car list)) (list-atoms (cdr list))))
  ))

> (list-atoms ())
NIL
> (list-atoms '(a))
(A)
> (list-atoms '(a b))
(A B)
> (list-atoms '(a (b c)))
(A B C)
> (list-atoms '((a b) c (d (e (g) f))))
(A B C D E G F)
> (list-atoms '(() a))
NIL A

runner% cat list-atoms.l
(defun list-atoms (list)
  (cond ((null list) nil)
        ((null (car list)) (list-atoms (cdr list)))
        ((atom (car list)) (cons (car list) (list-atoms (cdr list))))
        (t (append (list-atoms (car list)) (list-atoms (cdr list))))
  ))

> (list-atoms ())
NIL
> (list-atoms '(a))
(A)
> (list-atoms '(a b))
(A B)
> (list-atoms '(a (b c)))
(A B C)
> (list-atoms '((a b) c (d (e (g) f))))
(A B C D E G F)
> (list-atoms '(() a))
NIL A

runner% cat list-atoms.l
(defun list-atoms (list)
  (cond ((null list) nil)
        ((and (not (null (car list))) (atom (car list)))
         (cons (car list) (list-atoms (cdr list))))
        ((null (car list)) (list-atoms (cdr list)))
        (t (append (list-atoms (car list)) (list-atoms (cdr list))))
  ))

> ; note: how append and cons handle nil
   (append nil '(a) nil)
(A)
> (cons 'a nil)
(A)