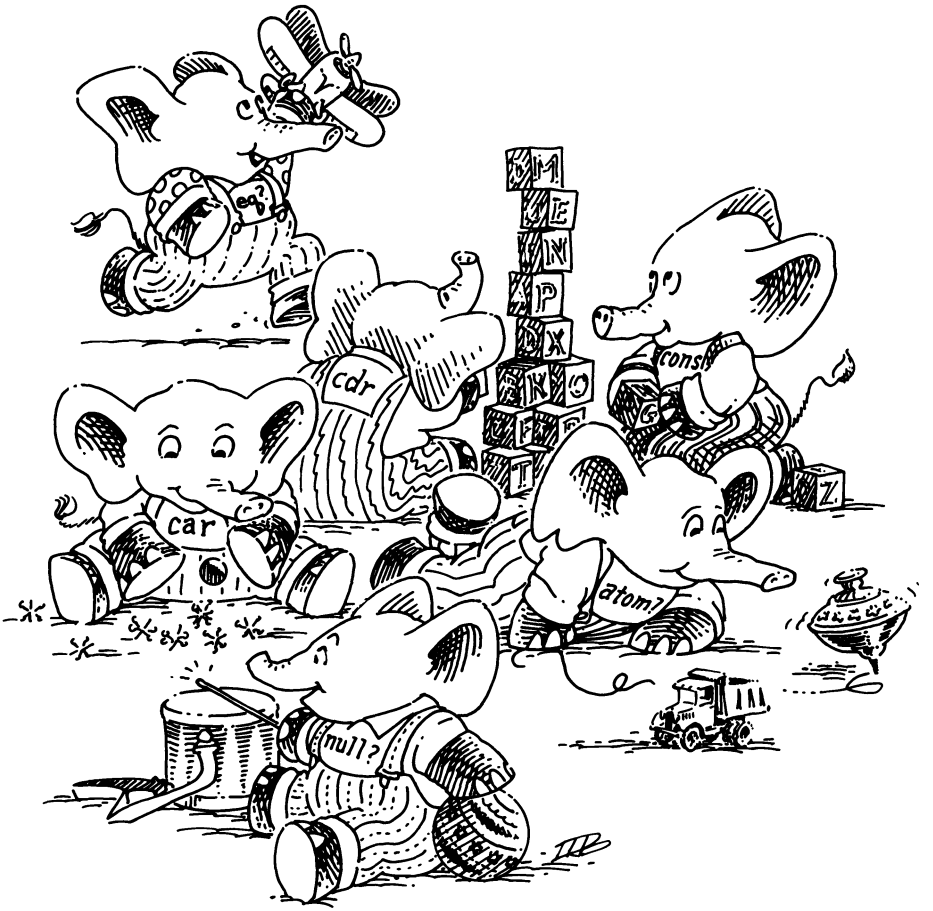
1. Toys



Is it true that this is an atom?
  atom[1]

Yes,
  because atom is a string of characters beginning with the letter a.

[1] L, S: `(quote atom)` or `'atom`
"L:" and "S:" are described in the preface.

Is it true that this is an atom?
  turkey

Yes,
  because turkey is a string of characters beginning with a letter.

Is it true that this is an atom?
  1492

Yes,
  because 1492 is a string of digits.

Is it true that this is an atom?
  u

Yes,
  because u is a string of one character, which is a letter.

Is it true that this is an atom?
  *abc$

Yes,
  because *abc$ is a string of characters beginning with a letter or special character other than a left "(" or right ")" parenthesis.

Is it true that this is a list?
  (atom)[1]

Yes,
  because (atom) is an atom enclosed by parentheses.

[1] L, S: `(quote (atom))` or `'(atom)`

Is it true that this is a list?
  (atom turkey or)

Yes,
  because it is a collection of atoms enclosed by parentheses.

Is it true that this is a list? (atom turkey) or	No, because these are actually two S-expressions not enclosed by parentheses. The first one is a list containing two atoms, and the second one is an atom.
Is it true that this is a list? ((atom turkey) or)	Yes, because the two S-expressions are now enclosed by parentheses.
Is it true that this is an S-expression? xyz	Yes, because all atoms are S-expressions.
Is it true that this is an S-expression? (x y z)	Yes, because it is a list.
Is it true that this is an S-expression? ((x y) z)	Yes, because all lists are S-expressions.
Is it true that this is a list? (how are you doing so far)	Yes, because it is a collection of S-expressions enclosed by parentheses.
How many S-expressions are in the list (how are you doing so far) and what are they?	Six, how, are, you, doing, so, and far.
Is it true that this is a list? (((how) are) ((you) (doing so)) far)	Yes, because it is a collection of S-expressions enclosed by parentheses.
How many S-expressions are in the list (((how) are) ((you) (doing so)) far) and what are they?	Three, ((how) are), ((you) (doing so)), and far.

Is it true that this is a list?
  ()

Yes,
  because it contains zero S-expressions enclosed by parentheses. This special S-expression is called the null (or empty) list.

Is it true that this is an atom?
  ()

No,
  because () is just a list.

Is it true that this is a list?
  (() () () ())

Yes,
  because it is a collection of S-expressions enclosed by parentheses.

What is the *car* of *l*
where *l* is the argument
  (a b c)

a,
  because a is the first atom of this list.

What is the *car* of *l*
where
  l is ((a b c) x y z)

(a b c),
  because (a b c) is the first S-expression of this non-empty list.

What is the *car* of *l*
where *l* is hotdog

No answer.
  You cannot ask for the *car* of an atom.

What is the *car* of *l*
where *l* is ()

No answer.[1]
  You cannot ask for the *car* of the empty list.

[1] L: `nil`

The Law of Car

The primitive *car* is defined only for non-empty lists.

What is the *car* of *l* where *l* is (((hotdogs)) (and) (pickle) relish)	((hotdogs)), read as: "The list of the list of hotdogs." ((hotdogs)) is the first S-expression of *l*.
What is (*car l*) where *l* is (((hotdogs)) (and) (pickle) relish)	((hotdogs)), because (*car l*) is another way to ask for "the *car* of the list *l*."
What is (*car* (*car l*)) where *l* is (((hotdogs)) (and))	(hotdogs).
What is the *cdr* of *l* where *l* is (a b c) Note: "cdr" is pronounced "could-er."	(b c), because (b c) is the list *l* without (*car l*).
What is the *cdr* of *l* where *l* is ((a b c) x y z)	(x y z).
What is the *cdr* of *l* where *l* is (hamburger)	().
What is (*cdr l*) where *l* is ((x) t r)	(t r), because (*cdr l*) is just another way to ask for "the cdr of the list *l*."
What is (*cdr a*) where *a* is hotdogs	No answer. You cannot ask for the *cdr* of an atom.

What is (*cdr l*)
where *l* is ()

No answer.[1]
You cannot ask for the *cdr* of the null list.

[1] L: `nil`

The Law of Cdr

The primitive *cdr* is defined only for non-empty lists. The *cdr* of any non-empty list is always another list.

What is (*car* (*cdr l*))
where
l is ((b) (x y) ((c)))

(x y),
because ((x y) ((c))) is (*cdr l*), and (x y) is the *car* of (*cdr l*).

What is (*cdr* (*cdr l*))
where
l is ((b) (x y) ((c)))

(((c))),
because ((x y) ((c))) is (*cdr l*), and (((c))) is the *cdr* of (*cdr l*).

What is (*cdr* (*car l*))
where
l is (a (b (c)) d)

No answer,
since (*car l*) is an atom, and *cdr* does not take an atom as an argument; see The Law of Cdr.

What does *car* take as an argument?

It takes any non-empty list.

What does *cdr* take as an argument?

It takes any non-empty list.

What is the *cons* of the atom *a* and the list *l*
where *a* is peanut
and
l is (butter and jelly)
This can also be written "(*cons a l*)".
Read: "*cons* the atom *a* onto the list *l*."

(peanut butter and jelly),
because *cons* adds an atom to the front of a list.

What is the *cons* of *s* and *l*
where *s* is (banana and)
and
l is (peanut butter and jelly)

((banana and) peanut butter and jelly),
because *cons* adds any S-expression to the front of a list.

What is (*cons s l*)
where
s is ((help) this)
and
l is (is very ((hard) to learn))

(((help) this) is very ((hard) to learn)).

What does *cons* take as its arguments?

cons takes two arguments:
the first one is any S-expression;
the second one is any list.

What is (*cons s l*)
where
s is (a b (c))
and
l is ()

((a b (c))),
because () is a list.

What is (*cons s l*)
where *s* is a
and *l* is ()

(a).

What is (*cons s l*)
where
s is ((a b c))
and
l is b

No answer,[1]
since the second argument *l* must be a list.

[1] In practice, (**cons** α β) works for all values α and β, and
(**car** (**cons** α β)) = α
(**cdr** (**cons** α β)) = β.

What is (*cons s l*)
where *s* is a
and *l* is b

No answer.
Why?

The Law of Cons

The primitive *cons* takes two arguments. The second argument to *cons* must be a list. The result is a list.

What is (*cons s* (*car l*))
where *s* is a
and
l is ((b) c d)

(a b).
Why?

What is (*cons s* (*cdr l*))
where *s* is a
and
l is ((b) c d)

(a c d).
Why?

Is it true that the list *l* is the null list
where *l* is ()

Yes,
because it is the list composed of zero S-expressions.
This question can also be written:
(*null? l*).

What is (*null?*[1] (**quote** ()))

True,
because (**quote** ())[1] is a notation for the null list.

[1] L: `null`

[1] L: Also `()` and `'()`.
S: Also `'()`.

Is (*null? l*) true or false
where
l is (a b c)

False,
because *l* is a non-empty list.

Is (*null? a*) true or false
where
a is spaghetti

No answer,[1]
because you cannot ask *null?* of an atom.

[1] In practice, (null? α) is false for everything, except the empty list.

The Law of Null?

The primitive *null?* is defined only for lists.

Is it true or false that *s* is an atom
where *s* is Harry

True,
because Harry is a string of characters beginning with a letter.

Is (*atom?*[1] *s*) true or false
where
s is Harry

True,
because (*atom? s*) is just another way to ask "Is *s* is an atom?"

[1] L: `(defun atom? (x)`
`(not (listp x)))`
S: `(define atom?`
`(lambda (x)`
`(and (not (pair? x)) (not (null? x)))))`

Is (*atom? s*) true or false
where
s is (Harry had a heap of apples)

False,
since *s* is a list.

How many arguments does *atom?* take and what are they?

It takes one argument. The argument can be any S-expression.

Is (*atom?* (*car l*)) true or false where *l* is (Harry had a heap of apples)	True, because (*car l*) is Harry, and Harry is an atom.
Is (*atom?* (*cdr l*)) true or false where *l* is (Harry had a heap of apples)	False.
Is (*atom?* (*cdr l*)) true or false where *l* is (Harry)	False, because the list () is not an atom.
Is (*atom?* (*car* (*cdr l*))) true or false where *l* is (swing low sweet cherry oat)	True, because (*cdr l*) is (low sweet cherry oat), and (*car* (*cdr l*)) is low, which is an atom.
Is (*atom?* (*car* (*cdr l*))) true or false where *l* is (swing (low sweet) cherry oat)	False, since (*cdr l*) is ((low sweet) cherry oat), and (*car* (*cdr l*)) is (low sweet), which is a list.
True or false: *a1* and *a2* are the same atom where *a1* is Harry and *a2* is Harry	True, because *a1* is the atom Harry and *a2* is the atom Harry.
Is (*eq?*[1] *a1 a2*) true or false where *a1* is Harry and *a2* is Harry	True, because (*eq? a1 a2*) is just another way to ask, "Are *a1* and *a2* the same non-numeric atom?"
Is (*eq? a1 a2*) true or false where *a1* is margarine and *a2* is butter	False, since *a1* and *a2* are different atoms.

[1] L: eq

How many arguments does *eq?* take and what are they?

It takes two arguments. Both of them must be non-numeric atoms.

Is (*eq? l1 l2*) true or false
where *l1* is ()
and
l2 is (strawberry)

No answer,[1]
() and (strawberry) are lists.

[1] In practice, lists may be arguments of **eq?**. Two lists are **eq?** if they are the same list.

Is (*eq? n1 n2*) true or false
where *n1* is 6
and
n2 is 7

No answer,[1]
6 and 7 are numbers.

[1] In practice, some numbers may be arguments of **eq?**.

The Law of Eq?

The primitive *eq?* takes two arguments. Each must be a non-numeric atom.

Is (*eq?* (*car l*) *a*) true or false
where
l is (Mary had a little lamb chop)
and
a is Mary

True,
because (*car l*) is the atom Mary, and the argument *a* is also the atom Mary.

Is (*eq?* (*cdr l*) *a*) true or false
where
l is (soured milk)
and
a is milk

No answer.
See The Laws of Eq? and Cdr.

Is (*eq?* (*car l*) (*car* (*cdr l*))) true or false
where
l is (beans beans we need jelly beans)

True,
because it compares the first and second atoms in the list.

⟹ Now go make yourself a peanut butter and jelly sandwich. ⟸

This space reserved for

JELLY STAINS!