

Introduction to LISP

LISP is actually a very simple language. Here is enough information about LISP to get started:

A. Syntactic Elements:

LISP is made up of two basic parts:

1. atoms — numbers like 2, 3, or 5.6 and identifiers like A or FOO.
2. lists — lists are represented by a list of elements enclosed in parentheses. For example, (A), (A B C), and () are lists (the last one is the empty list). (A (B C) D) is also a list; the only thing different about it is that one of its elements is another list.

LISP has two special, pre-defined atoms:

1. NIL — This is a name for the empty list. In LISP this is also the value FALSE.
2. T — This is a LISP value for TRUE. In LISP, anything that is not the empty list (NIL) has the value TRUE, but T is pre-defined.

B. Functions:

1. Lisps function calls use a different syntax than that of the languages you are used to. If `foo` is the name of a function with three parameters, then the function call looks like this:

```
(foo parm1 parm2 parm3)
```

2. LISP's function definitions look like this:

```
(defun foo (parm1 parm2 parm3)
  (body1)
  (body2)
  (body3) )
```

where `body1`, `body2`, and `body3` are function bodies. LISP first evaluates all the parameters in the function call, then replaces the parameter names in the bodies with the values it got by evaluating the parameters, then executes the function bodies one at a time. It returns the value it got from executing the last function body as the result of the function.

3. Here are some of the built-in LISP functions:
 - a. `CAR` — Takes one parameter which should be a list. Returns the first element in the list.
 - b. `CDR` — Takes one parameter which should be a list. Returns the list without its first element.
 - c. `CONS` — Takes two parameters, the second of which should be a list. Returns a new list which is the old second parameter with a new first element which is the first parameter.
 - d. `+` — Takes two parameters which should be numbers. Returns their sum.
 - e. `*` — Takes two parameters which should be numbers. Returns their product.

- f. `QUOTE` — Takes one parameter. Returns the literal representation of the parameter (essentially, it prevents LISP from evaluating the parameter).
- g. `SET` — Takes two parameters, the first of which should be a quoted identifier name and the second of which is a value. It returns the value but it also has the side effect of binding that value to the identifier name.
- h. `NULL` — Takes one parameter. Returns `T` if the parameter is the empty list and `NIL` if not.

C. Conditional:

LISP has a special function that is used to make decisions. Its form is:

```
(cond
  (expr1 expr2)
  (expr3 expr4)
  (expr5 expr6) )
```

Each pair of expressions is called a clause. When LISP executes a `cond` function, it evaluates the first expression in the first clause, in this case `expr1`. If its value is `TRUE`, LISP evaluates `expr2` and returns that value as the result of the `cond` function. If it is `FALSE`, LISP moves to the next clause and repeats the cycle. LISP continues to do this until it finds a clause with a first expression that evaluates to `TRUE` or until it runs out of clauses. In the latter case, `cond` returns a `NIL`.

If you want to have a “default” or “otherwise” clause in a LISP `cond` function, the last clause should look like this:

```
(T expr7) )
```