

CHAPTER 1

GETTING STARTED

Your goal for this chapter is to learn how to access your system's COMMON LISP and how to exit from it. Although this may not seem like much, it is obviously very important. It is also very dependent on the particular system you are using, so you will have to get much of the information from a manual, a teacher, a consultant, or a friend.

The first problem is to *log onto* your computer system. This might involve simply turning on your microcomputer or it might require typing in some accounting information.

If you are using a LISP machine or have "booted" a microcomputer with a COMMON LISP disk, you may already be talking to COMMON LISP. Otherwise, you will have to access it. This might require just typing LISP, CL, some abbreviation of the implementation of COMMON LISP that you are using, or it might require first retrieving the COMMON LISP system.

Once you have started your COMMON LISP, you are ready to interact with it. We say that you are about to interact with the *top-level LISP listener*, or simply the *top level* of LISP.

Most COMMON LISP listeners will tell you they are waiting for input by printing a *prompt*. This can be a greater-than symbol, a question mark, an arrow, a colon, or something else. You are now supposed to type something to LISP called a *symbolic expression*, or *S-expression*.

We will get into great detail about what an S-expression is, but for now, let's use small numerals, like 3.

When you type an S-expression to LISP (remember to end each entry by pressing the CARRIAGE RETURN key), LISP will perform the following sequence of actions:

1. It will *read* your S-expression.
2. It will interpret your S-expression as the *printed representation* of a *form*—a LISP *object* intended to be evaluated.
3. It will *evaluate* the form as some other (or perhaps as the same) *value object*.
4. It will choose a printed representation for the value object.
5. It will *print* the printed representation it has chosen.

Because of this sequence, LISP listeners are also called *read-eval-print* loops (combining steps 1 and 2 into the read step, and steps 4 and 5 into the print step).

After printing each value, the LISP listener will again print a prompt (or not, if it's one of those COMMON LISPs that don't use prompts) and will wait for your next S-expression. That's all there is to using LISP: you type the printed representation of a form; LISP evaluates it and types back a printed representation of the value of the form.

For our first example, the S-expression we will enter is the arabic numeral 3. Notice that this is only one of the printed representations we use in our daily lives for the number 3. Another common printed representation we use for 3 is the roman numeral III. I mention this not because COMMON LISP uses roman numerals, but to point out that the distinction between printed representations of objects and the objects themselves is one you are already familiar with. Anyway, LISP interprets the numeral 3 as representing the number 3 and evaluates that form (that is, the numeric object 3). In LISP, numbers evaluate to themselves, so 3 evaluates to 3. LISP then must choose a printed representation for 3 and, in fact, chooses the arabic numeral 3 and prints that.

In this text, I will show a sample LISP interaction as:

```
> 3
3
```

The `>` is what I will use for the LISP prompt, and it is followed by an S-expression as you would type it to LISP. The line after that shows LISP's response.

In some LISPs, when you make a mistake, or when you make certain mistakes, LISP will enter a *debugger* (sometimes called a *break loop* or *break package*). The debugger is a LISP listener, just like the top level, except that some special commands are available to obtain information relevant to figuring out your error. We will look at this in more detail in later chapters. If you make a mistake while in the debugger, you may (depending on the implementation of COMMON LISP you are using) get into another debugger. The first debugger will remember the information relevant to your first mistake; the second one will have information relevant to your second mistake. This can go on for many levels of nested debuggers.

The debugger is recognizable because it uses a different prompt. For example, Kyoto COMMON LISP's (KCL's) top-level prompt is `>`, while its debugger prompt is `>>`. To get out of KCL's debugger, type `:q`. If you are several levels down in debuggers, you may have to do this repeatedly, or your COMMON LISP might have a single command to jump all the way back to the top level. For example Lucid's COMMON LISP uses `:a` to return to the top-level listener from any debugger level.

If the LISP you're using has a debugger, you can often force your way into it by typing the appropriate *interrupt key*. This may be the `BREAK`, `RUB`, or `DEL` key, or it may be some control character such as `CTRL-C` (this means typing the `c` key while holding down the `CTRL` key). Sometimes, for the interrupt key to work, it must be struck before any other character is typed on the line, and sometimes it must be typed more than once in succession.

Having returned to the top-level listener, you may want to terminate your LISP session. The way to get out of COMMON LISP varies with different implementations. To leave KCL, type `(bye)` followed by a carriage return. (It is important to type the parentheses as shown.) Other implementations may use `(exit)`, `(system: exit)`, `CTRL-D`, or something else.

Finally, you need to know how to *log off* your computing system. As was the case for the other system-dependent information discussed in this chapter, you must find out how to do that from your manual, your teacher, a consultant, or a friend.

Exercises

- 1.1 (i) What is the procedure for getting into your LISP? Find out and write it here:

- 1.2 (i) What is the procedure for getting out of your LISP? Find out and write it here:

- 1.3 (i) Get into LISP. What is the top-level listener's prompt? Write it here: _____

- 1.4 (d) Get out of LISP and log off.

- 1.5 (r) Get back into LISP. Enter the numeral 3 and a carriage return. Note that LISP types 3 back and issues a prompt. Try 5 this time. Log off.

- 1.6 (i) Get back into LISP. Does it have an interrupt key? If so, write it here: _____ and get into the debugger.

- 1.7 (i) What is your debugger's first-level prompt? Write it here: _____

- 1.8 (i) How do you get out of the debugger? Write it here: _____

Do it! Are you back to the top level? _____

- 1.9 (i) Try going at least three levels deep in the debugger. Does the prompt change again? Write the answer here: _____
- 1.10 (r) While in the debugger, try typing a small numeral to LISP. LISP should echo it.
- 1.11 (i) How do you go back up a single level in the debugger? Write it here:_____ Do it.
- 1.12 (i) How do you go all the way to the top-level listener from deep in the debuggers? Write the answer here:_____ Do it.
- 1.13 (d) Exit LISP and log off your system.
- 1.14 (u) Copy all the answers you wrote here to the appropriate blanks in Appendix B.2.