CHAPTER 1

LISP

LISP is an interesting programming language, and the ideas involved in building a LISP interpreter are equally interesting [McC79]. This book contains an introduction to LISP and it also contains the data structure details and the explicit code for a working LISP interpreter.

LISP is a programming language with unique features. It is conceptually interactive. Input commands are given one by one and the associated result values are printed out. LISP is an applicative language, meaning that it consists mainly of functional application commands. Besides functional application, there are forms of assignment commands and conditional commands written in functional form. In general, iteration is replaced by recursion.

The data values on which a LISP function may operate includes real numbers. Thus, an expression like 1.5 + 2 is a LISP statement, which means: type out the result of applying + to the arguments 1.5 and 2. In LISP, function application statements are always written in prefix form, for example, +(1.5, 2). Moreover, rather than writing $f(x, y)$ to indicate the result of the function f applied to the arguments x and y, we write $(f\,x\,y)$ in LISP, so (+ 1.5 2) is the LISP form for 1.5 + 2. Finally, functions in LISP are usually specified by identifier names rather than special symbols. Thus the correct way to compute 1.5 + 2 in LISP is to enter the expression `(PLUS 1.5 2)`, which will, indeed, cause 3.5 to be printed out. An expression such as `(PLUS 1.5 2)` is called a *function call expression*. LISP functions can also operate on *lists of objects*; indeed the acronym LISP is derived from the phrase LISt Processing.

LISP is commonly implemented with an interpreter program called the LISP Interpreter. This program reads LISP expressions that are entered as input and evaluates them and prints out the results. Some expressions specify that state-changing side-effects also occur. We shall describe below how a particular LISP interpreter is constructed at the same time that LISP itself is described.

There are a variety of dialects of LISP, including extended forms, which have enriched collections of functions and additional datatypes. We shall focus on the common core of LISP, but some definitions given here are not universal, and in a few cases they are unique to the version of LISP presented herein (GOVOL). The GOVOL dialect of LISP presented here is similar to the original LISP 1.5 [MIT62]; it is not as complex as the current most frequently used varieties of LISP, but it contains all the essential features of these more complex varieties, so that what you learn in this book will be immediately applicable for virtually every LISP dialect. (Look up the programming language *Jovial* to learn the meaning of GOVOL.)

G. D. Knott, *Interpreting LISP*, DOI 10.1007/978-1-4842-2707-7_1