The First Ten Commandments

The First Commandment

When recurring on a list of atoms, *lat*, ask two questions about it: (*null? lat*) and **else**.
When recurring on a number, *n*, ask two questions about it: (*zero? n*) and **else**.
When recurring on a list of S-expressions, *l*, ask three question about it: (*null? l*), (*atom?* (*car l*)), and **else**.

The Second Commandment

Use *cons* to build lists.

The Third Commandment

When building a list, describe the first typical element, and then *cons* it onto the natural recursion.

The Fourth Commandment

Always change at least one argument while recurring. When recurring on a list of atoms, *lat*, use (*cdr lat*). When recurring on a number, *n*, use (*sub1 n*). And when recurring on a list of S-expressions, *l*, use (*car l*) and (*cdr l*) if neither (*null? l*) nor (*atom? (car l)*) are true.

It must be changed to be closer to termination. The changing argument must be tested in the termination condition:

when using *cdr*, test termination with *null?* and

when using *sub1*, test termination with *zero?*.

The Fifth Commandment

When building a value with ✣ ,always use 0 for the value of the terminating line, for adding 0 does not change the value of an addition.

When building a value with ×, always use 1 for the value of the terminating line, for multiplying by 1 does not change the value of a multiplication.

When building a value with *cons*, always consider () for the value of the terminating line.

The Sixth Commandment

Simplify only after the function is correct.

The Seventh Commandment

Recur on the *subparts* that are of the same nature:

- On the sublists of a list.
- On the subexpressions of an arithmetic expression.

The Eighth Commandment

Use help functions to abstract from representations.

The Ninth Commandment

Abstract common patterns with a new function.

The Tenth Commandment

Build functions to collect more than one value at a time.

The Next Ten Commandments

The Eleventh Commandment

Use additional arguments when a function needs to know what other arguments to the function have been like so far.

The Twelfth Commandment

Use (**letrec** ...) to remove arguments that do not change for recursive applications.

The Thirteenth Commandment

Use (**letrec** ...) to hide and to protect functions.

The Fourteenth Commandment

Use (**letcc** ...) to return values abruptly and promptly.

The Fifteenth Commandment

Use (**let** ...) to name the values of repeated expressions in a function definition if they may be evaluated twice for one and the same use of the function. And use (**let** ...) to name the values of expressions (without **set!**) that are re-evaluated every time a function is used.

The Sixteenth Commandment

Use (**set!** ...) only with names defined in (**let** ...)s.

The Seventeenth Commandment

Use (**set!** x ...) for (**let** ((x ...)) ...) only if there is at least one (**lambda** ... between it and the (**let** ...), or if the new value for x is a function that refers to x.

The Eighteenth Commandment

Use (**set!** x ...) only when the value that x refers to is no longer needed.

The Nineteenth Commandment

Use (**set!** ...) to remember valuable things between two distinct uses of a function.

The Twentieth Commandment

When thinking about a value created with (**letcc** ...), write down the function that is equivalent but does not forget. Then, when you use it, remember to forget.