```
;;;; The Little Schemer, 4ª ed., adaptado para Common Lisp.
;;;; Abrantes Araújo Silva Filho
;;;; abrantesasf@gmail.com

;;; LEIS e MANDAMENTOS

;; ------------------------------------│----------------------------------------
;; As 8 Leis:                          │ Os 10 Mandamentos
;; ------------------------------------│----------------------------------------
;; CAR (car l)                         │ 1) RECORRÊNCIA
;; deinido apenas para listas e retor- │ Ao recorrer numa lat, pergunte:
;; na a primeira s-exp ou NIL.         │   - (null lat)
;;                                     │   - else
;; CDR (cdr l)                         │ Ao recorrer num número, pergunte:
;; definido apenas para listas e re-   │   - (zero n)
;; torna uma lista sem (car l) ou NIL. │   - else
;;                                     │ Ao recorrer numa lista, pergunte:
;; CONS (cons s l)                     │   - (null l)
;; adiciona uma s-exp como o primeiro  │   - (atom (car l))
;; elemento de uma lista.              │   - else
;;                                     │
;; NULL (null s)                       │ 2) CONSTRUÇÃO DE LISTAS - I
;; definido para qualquer s-exp e re-  │ Sempre use CONS para construir listas.
;; torna T se objeto é nulo (uma lista │
;; vazia ou NIL) ou NIL casa não.      │ 3) CONSTRUÇÃO DE LISTAS - II
;;                                     │ Descreva o priemiro elemento típico e
;; EQ (eq x y)                         │ então faça o CONS desse elemento com a
;; retorna T se os objetos x e y são   │ recursão natural.
;; implementacionalmente idênticos.    │
;; Não é confiável para números, ca-   │ 4) NA RECURSÃO, ALTERE UM ELEMENTO
;; racteres ou listas. Pode ser utili- │ Sempre altere pele menos um argumento
;; zado para symbols.                  │ enquanto ocorrer a recursão:
;;                                     │ Em uma lat:
;; EQL (eql x y)                       │   - (cdr lat)
;; retorna T se os objetos x e y são   │ Em um número:
;; CONCEITUALMENTE os mesmos (mesmo    │   - (sub1 n), (add1 n), etc.
;; data type e valor para números e    │ Em uma lista, sempre que (null l) e
;; caracteres; não usar em listas).    │ (atom (car l)) forem falsos:
;;                                     │   - (car l)
;; EQUAL (equal x y)                   │   - (cdr l)
;; como EQ, mas serve para listas e    │ A alteração deve tornar o problema mais
;; e outras estruturas.                │ próximo do fim, e o novo argumento deve
;;                                     │ ser testado na condição de terminação:
;; EQUALP (equalp x y)                 │ Ao usar CDR, teste a terminação com:
;; como EQUAL mas independe do case da │   - NULL
;; letra e números são iguais se forem │ Ao usar SUB1/ADD1, teste a termianção:
;; o mesmo valor matemático.           │   - ZERO
;;                                     │
;;                                     │ 5) CONSTRUÇÃO COM +, x, CONS
;;                                     │ Ao construir um valor com +, sempre use
;;                                     │ 0 (zero) como valor da linha terminal.
;;                                     │ Ao construir um valor com x, sempre use
;;                                     │ 1 (um) como valor de linha terminal.
;;                                     │ Ao construir com CONS, considere usar
;;                                     │ () como valor da linha terminal.
;;                                     │
;;                                     │ 6) SIMPLIFIQUE POSTERIORMENTE
;;                                     │ Simplifique o código apenas depois da
;;                                     │ função estar correta.
;;                                     │
;;                                     │ 7) RECORRÊNCIA NATURAL
;;                                     │ Sempre faça a recorrência em subpartes
;;                                     │ que são de mesma natureza:
;;                                     │   - em sublistas de uma lista
```

```
65 ;;                                    |   - em subexpressões de uma expressão
66 ;;                                    |     aritmética
67 ;;                                    |
68 ;;                                    | 8) ABSTRAIA COM HELP FUNCTIONS
69 ;;                                    | Use help functions para abstração.
70 ;;                                    |
71 ;;                                    | 9) ABSTRAIA PADRÕES COMUNS COM FUNÇÕES
72 ;;                                    | Ao encontrar um padrão, crie uma função
73 ;;                                    | e abstraia.
74 ;;                                    |
75 ;;                                    | 10) MAIS DE UM VALOR AO MESMO TEMPO
76 ;;                                    | Construa funções que coletam mais de um
77 ;;                                    | valor ao mesmo tempo.
78 ;; ------------------------------------------------------------------------------
end
```