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

;;; Chapter 1: Toys

;; Atoms, Lists e S-expressions:
;; -----------------------------
;; Atoms: (atom <s-expression>) , (atom? <s-expression>)
;;        - seqüência de caracteres, dígitos e caracteres especiais
;;          (exceto parênteses)
;;
;; Lists: (listp l) , (list? l)
;;        - coleção de atoms (ou outras s-expressions) entre parênteses
;;          (pode ser vazia () ou NIL)
;;
;; S-expressions: todo Atom
;;                toda List

;; The Law of CAR:
;; ---------------
;; car: (car l)
;;      - definido apenas para LISTAS
;;      - retorna a PRIMEIRA S-expression, um atom ou uma lista, ou retorna
;;        NIL se a lista estiver vazia

;; The Law of CDR:
;; ---------------
;; cdr: (cdr l)
;;      - definido apenas para LISTAS
;;      - retorna uma LISTA sem a primeira s-expression, ou retorna NIL
;;        se a lista estiver vazia

;; The Law of CONS:
;; ----------------
;; cons: (cons s-exp l)
;;       - adiciona uma s-expression (atom ou lista) na frente de outra lista
;;       - o segundo argumento deve obrigatoriamente ser uma lista

;; The Law of NULL?
;; ----------------
;; null: (null s-exp) , (null? s-exp)
;;       - definido para QUALQUER s-expression
;;       - retorna T se o objeto for nulo, ou NIL de não for nulo (na prática
;;         tudo é falso - NIL - exceto para a lista vazia)

;; The Laws of EQUALITY:
;; ---------------------
;; (abaixo, x e y são objetos)
;;
;; eq: (eq x y) , (eq? x y)
;;     - retorna T se os objetos são os mesmos, IMPLEMENTACIONALMENTE IDÊNTICOS
;;     - PERIGOSO: depende de como os data types foram implementados e,
;;       portanto, a comparação NÃO É CONFIÁVEL: Em geral:
;;         - NÃO USE PARA NÚMEROS OU CARACTERES
;;         - NÃO USE PARA LISTAS
;;         - Pode ser usado para SYMBOLS cujo print seja igual
;;
;; eql: (eql x y) , (eql? x y)
;;      - retorna T se os objetos são os mesmos, CONCEITUALMENTE os mesmos
```

```
;;      - SEGURO: como não depende da implementação e, sim, do "conceito" do
;;        objeto, a comparação É CONFIÁVEL. Retorna T se:
;;          - eq retornar T
;;          - são números do MESMO DATA TYPE e de MESMO VALOR
;;          - são caracteres do MESMO DATA TYPE e representam o MESMO CARACTERE
;;
;; equal: (equal x y) , (equal? x y)
;;        - afrouxa a discriminação de EQL e considera listas equivalentes se
;;          tiverem a mesma estrutura e conteúdo, recursivamente. Retorna T se:
;;            - são SYMBOLS que são eq
;;            - são números ou caracteres que são eql
;;            - são CONSES onde os dois CAR e os dois CDR são equal
;;            - são ARRAYS que são eq
;;            - são strings ou bit vectors que são eql, elemento a elemento
;;            - são pathnames com todos os componentes equivalentes
;;            - são outras coisas (structures, hash-tables, instances...) eq
;;
;; equalp: (equalp x y) , (equalp? x y)
;;         - afrouxa a discriminação de EQUAL. Retorna T se:
;;             - caracteres são os mesmo, INDEPENDENTE DO CASE
;;             - números representam o mesmo VALOR MATEMÁTICO
```