Trabalho 2 de PLP: Lisp

Felipe Sampaio de Souza, RA: 619523
Marcio Lima Inácio, RA: 587265

1 Código fonte e documentação

```
; Dada uma lista L, constrói uma lista com apenas os elementos atômicos,
; inclusive os das sublistas.
(defun desparentize (L)
  (do ( (i 0 (+ i 1)) (res ()) )
    ( (= i (length L)) (reverse res) )
    (cond
      ; Se L[i] for átomo, adiciona a res
      ( (atom (nth i L)) (setq res (cons (nth i L) res)) )
      ; Se L[i] for lista, aplica desparentize nela e dá append com res
      ( (listp (nth i L)) (setq res (append (reverse (desparentize (nth i L))) res)) )
    )
  )
)

; Dadas duas listas L1 e L2, retira de L1 os elementos que não aparecem em L2.
(defun tira_nao_comuns (L1 L2)
  (cond ((null L1) nil)
        ((null L2) nil)
        (t (cond ((member (car L1) L2) (cons (car L1) (tira_nao_comuns (cdr L1) L2)))
                 (t (tira_nao_comuns (cdr L1) L2))))))

; Conta o número de ocorrências do elemento x na lista L
(defun conta_atomos (x L)
  (cond ((null L) 0)
        ((equal x (car L)) (+ 1 (conta_atomos x (cdr L))))
        (t (conta_atomos x (cdr L)))))

; Retira da lista L todas as ocorrências do elemento x
(defun retira_atomos (x L)
  (cond ((null L) nil)
        ((equal x (car L)) (retira_atomos x (cdr L)))
        (t (cons (car L) (retira_atomos x (cdr L))))))

; Dada uma lista L, monta uma lista de pares formados por um elemento da lista L
; e o número de vezes que ele aparece em L.
(defun monta_pares (L)
  (cond ((null L) nil)
        (t (cons (list (car L) (conta_atomos (car L) L)) (monta_pares (retira_atomos (car L) L))))))

; Recebe duas listas, L1 e L2, e retorna uma lista de pares (Elemento N)
; Onde Elemento éum atomo que aparece tanto em L1 quanto em L2
; E N éo número de ocorrência desse atomo nas duas listas
(defun conta_elementos (L1 L2)
  (setq L1 (desparentize L1))
  (setq L2 (desparentize L2))
  (setq L1 (tira_nao_comuns L1 L2))
  (setq L2 (tira_nao_comuns L2 L1))
  (monta_pares (append L1 L2)))

; Fornece a interface para o usuário inserir as listas em que deseja aplicar
; a função conta_elementos
(defun main ()
  (write-line "Digite a primeira lista: ")
  (setq L1 (read))
  (write-line "Digite a segunda lista: ")
  (setq L2 (read))
  (write-line "A lista de pares é: ")
  (write (conta_elementos L1 L2)))
```

```
; Inicializa o programa
(main)
```

2 Execução

```
Digite a primeira lista:
(a b Z (a x) () (5 z) () (par c d))
Digite a segunda lista:
(a 4.6 x () x b () (par c d))
A lista de pares é:
((A 3) (B 2) (X 3) (NIL 4) (PAR 2) (C 2) (D 2))
```

Figure 1: Primeiro exemplo de execução

```
Digite a primeira lista:
(2 () (h e l o i s (a)) () (plp t1))
Digite a segunda lista:
((plp X) ((())) a (((2))))
A lista de pares é:
((2 2) (NIL 3) (A 2) (PLP 2))
```

Figure 2: Segundo exemplo de execução