PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO – 2017-1

Profa. Heloisa - Segundo Trabalho - **DATA DE ENTREGA: 29/06/2017**

==========

Escreva um programa em LISP para, dadas duas listas *L1* e *L2* que contém elementos de qualquer tipo, possivelmente com repetições e sublistas, construir uma lista que mostre quantas vezes cada elemento atômico (átomo, número, lista vazia) aparece nas duas listas dadas, inclusive nas sublistas. A lista resultante deve conter pares de elementos (na forma de lista) sendo o primeiro elemento do par um elemento atômico que aparece nas duas listas dadas e o segundo elemento do par, o número de vezes que esse elemento aparece nas duas listas. Os elementos que aparecem em apenas uma das duas listas dadas não devem ser contados e não aparecem na lista final (ver exemplo do número 5, átomo z e número 4.6 nas listas abaixo).

Por exemplo, dadas as listas:

L1 = (a b z (a x) () (5 z) () (c d)) e L2 = (a 4.6 x () x b () (c,d))

Deve ser construída a lista

((a 3) (b 2) (x 3) (() 4) (c 2) (d 2)).

Sugestão: Construir as listas contendo só elementos atômicos (desparentizar) e remover os elementos que aparecem em só uma das listas. Depois, fazer o *append* dessas duas listas e construir a lista de pares.

Para isso devem ser definidos:

- Uma função (*conta_elementos L1 L2)* que recebe as listas *L1* e *L2* e constrói a lista de pares da forma solicitada, usando a função *desparentize* e possivelmente outras funções auxiliares, se for conveniente (obrigatório);
- Uma função (*desparentize L)* que, dada uma lista *L*, constrói uma lista com apenas os elementos atômicos, inclusive os das sublistas (obrigatório usar recursão);
- Uma função (*tira_nao_comuns L1 L2)* que, dadas duas listas *L1* e *L2*, retira de *L1* os elementos que não aparecem em *L2*. Essa função deve ser chamada duas vezes, repetindo a operação para retirar de *L2* os elementos que não aparecem em *L1*. Note que não basta calcular a interseção das duas listas, pois senão, as repetições em *L2* não aparecem na lista resultante (opcional);
- Um predicado (*monta_pares L)* que, dada uma lista de elementos *L*, sem sublistas, monta uma lista de pares formados por um elemento da lista *L* e o número de vezes que ele aparece em *L*;
- Outros predicados que achar necessário, se desejar subdividir as etapas da operação ou melhorar a entrada e saída.

Observações:

- Podem ser utilizados, se necessário, as funções e predicados pré-definidos de Lisp null, atom, cons, car, cdr, list, append, e outros;
- Os trabalhos podem ser feitos em duplas, que deverão ser as mesmas em todos os trabalhos;
- Usar OBRIGATORIAMENTE, CLISP para implementar o trabalho;
- Entregar, via Moodle (tarefa com envio de arquivo único):
  - Relatório (Arquivo txt, doc ou pdf) com listagem do código fonte, explicação do funcionamento de cada uma das funções definidas e resultados de execução de pelo menos dois exemplos;
  - Arquivo do lisp com o código fonte do trabalho.

- **DATA DE ENTREGA: 29/06/2017**