PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO – 2017-1

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

==========

Escreva um programa Prolog para, dadas duas listas *L1 e L2* que contém elementos de qualquer tipo, possivelmente com repetições, construir outra lista que mostre quantas vezes cada elemento atômico (átomo, número, lista vazia), ou estrutura, que não seja lista, aparece nas duas listas dadas, inclusive nas sublistas. A lista resultante deve conter pares de elementos sendo o primeiro elemento do par um elemento 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 não devem ser contados e não aparecem na lista final (ver exemplo do número 5, átomo z e número 4.6 nas listas abaixo). Variáveis também devem ser descartadas (ver exemplo da variável Z nas listas abaixo).

Por exemplo, dadas as listas:

L1 = [a, b, Z, [a, x], [], [5,z], [], par(c,d)] e L2 = [a, 4.6, x, [], x, b, [], par(c,d)]

Deve ser construída a lista

Lout = [[a, 3], [b, 2], [x, 3], [[], 4], [par(c,d), 2]].

Sugestão:

Desparentizar as duas listas (construir listas sem sublistas), remover as variáveis 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:

- Um predicado *conta_atomos(L1, L2, Lout)* que executa a operação solicitada (obrigatório);
- Um predicado principal para ler as duas listas dadas do teclado, e chamar o predicado *conta_atomos(L1,L2,Lout)*, que faz a operação principal. A lista resultante pode ser mostrada de qualquer forma (como variável na consulta ou com write) (obrigatório);
- Um predicado *desparentize(L, Lout)* que, dada uma lista *L*, constrói uma lista *Lout* sem sublistas, com apenas os elementos atômicos e estruturas que aparecem em todos os níveis, descartando as variáveis. As variáveis podem ser removidas pelo próprio predicado *desparentize* ou podem ser removidas depois, por outro predicado (opcional);
- Um predicado *tira_nao_comuns(L1,L2,L)* que, dadas duas listas *L1* e *L2*, retira de *L1* os elementos que não aparecem em *L2*. Esse predicado deve ser chamado 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(Lin, Lout)* que, dada uma lista de elementos *Lin*, sem sublistas, monta uma lista de pares formados por um elemento da lista *Lin* e o número de vezes que ele aparece em *Lin*;
- Outros predicados que achar necessário, se desejar subdividir as etapas da operação ou melhorar a entrada e saída.

Observações:

- Podem ser utilizados, se necessário, os predicados pré-definidos de Prolog is_list(L), atomic(X), var(X), compound(X), not(G), append(L1, L2,Lout), member(X,L), write(X), read(X) ;
- Os trabalhos podem ser feitos em duplas, que deverão ser as mesmas em todos os trabalhos;
- Usar OBRIGATORIAMENTE, SWI-PROLOG 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 um dos predicados definidos e resultados de execução de pelo menos dois exemplos;
  - Arquivo do prolog com o código fonte do trabalho.

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