Trabalho 1 (PLP - 2017)

Autores:
Alexandre Regali Seleghim RA: 551473
Thiago Bretas de Souza RA: 551899

O programa possui 6 predicados:
desparentize(Lista_in, Lista_out),
tira_nao_comuns(Lista1_in, Lista2_in, Lista_out),
conta(Elemento, Lista_in, Numero_ocorrencias),
remove_elemento(Elemento, Lista_in, Lista_out),
final(Lista_in, Lista_out),
conta_atomos(Lista1_in, Lista2_in, Lista_out).

DESPARENTIZE:
Desparentiza a lista de entrada removendo todas as sublistas e adicionando seus elementos a lista principal.
Exemplo:
desparentize([abacaxi, [cebola, maça, [morango]], [[[[manga]]]]], Lout)
Lout = [abacaxi, cebola, maça, morango, manga]

TIRA_NAO_COMUNS:
Tira todos os objetos da Lista 1 que não pertencem a Lista 2 (operação de intersecção).
Exemplo:
tira_nao_comuns([a, b, c, d], [a, j, c, k], Lout)
Lout = [a, c]

CONTA:
Conta todas as ocorrências de um certo elemento em uma lista.
Exemplo:
conta(juninho, [felipe, ana, juninho, pedro, juninho, lucas, ana], N)
N = 2

REMOVE_ELEMENTO:
Remove todas as ocorrências de um certo elemento em uma lista.
Exemplo:
remove_elemento(juninho, [felipe, ana, juninho, pedro, juninho, lucas, ana], Lout)
Lout = [felipe, ana, pedro, lucas, ana]

FINAL:
Cria uma lista contendo sublistas com o par de cada elemento de uma outra lista com o seu número de ocorrências. Os seguintes passos são seguidos:

-Conta o primeiro elemento da Lista_in
-Adiciona o par [Elemento, N] à Lista_out
-Remove todas as ocorrências de Elemento da Lista_in
-Repete recursivamente até que Lista_in fique vazia
-Imprime a Lista_out (write(Lout))

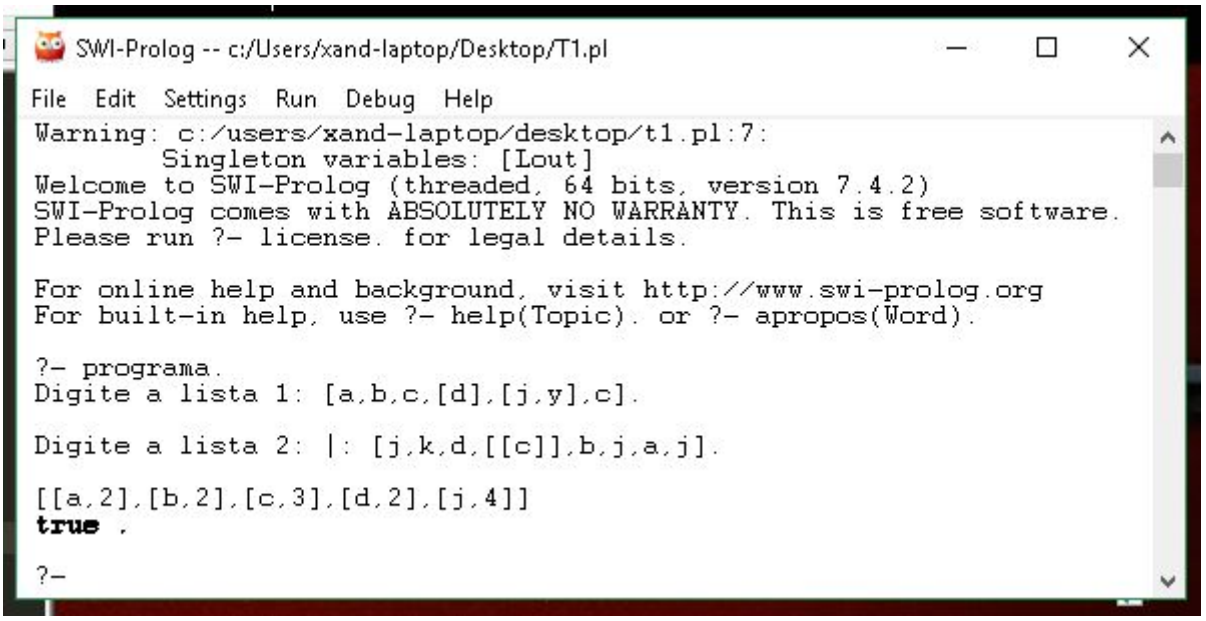
Exemplo:
final([mia, juninho, batman, mia, mia, batman], Lout)
Lout = [[mia, 3], [juninho, 1], [batman, 2]]

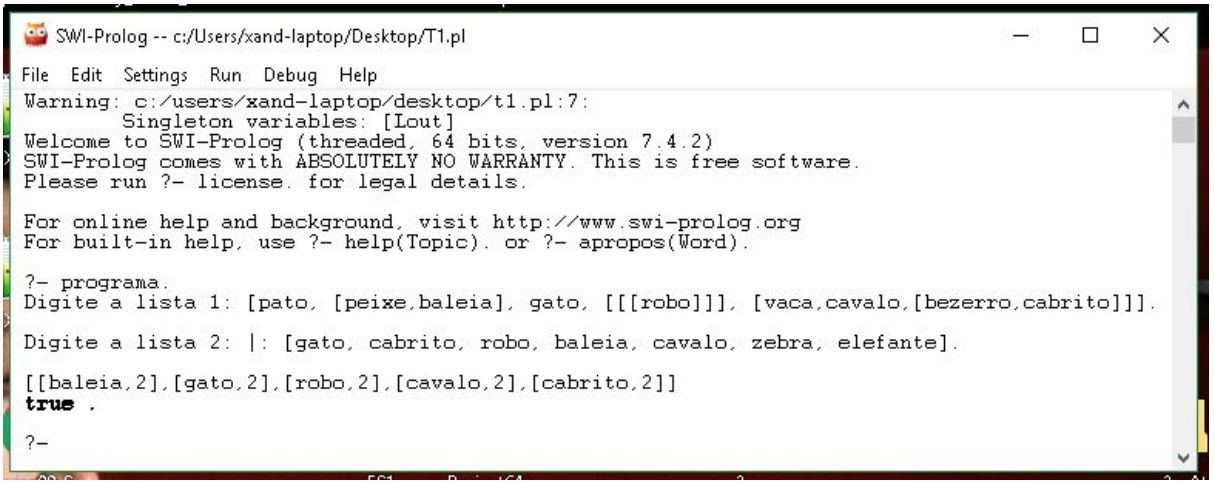
CONTA_ATOMOS:
Prepara as listas de entrada para depois serem usadas no predicado FINAL. Ele segue os seguintes passos:

-Desparentiza Lista 1
-Desparentiza Lista 2
-Tira não comuns da Lista 1 em relação à Lista 2
-Tira não comuns da Lista 2 em relação à Lista 1
-Junta as duas listas criando Lista 3 (append)
-Chama o predicado FINAL com a Lista 3

O programa principal está lendo Lista 1 e Lista 2 e depois chamará conta_atomos(Lista 1, Lista 2, Lista_out).

Exemplos:


SWI-Prolog -- c:/Users/xand-laptop/Desktop/T1.pl
File Edit Settings Run Debug Help
Warning: c:/users/xand-laptop/desktop/t1.pl:7:
Singleton variables: [Lout]
Welcome to SWI-Prolog (threaded, 64 bits, version 7.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?- programa.
Digite a lista 1: [a,b,c,[d],[j,y],c].
Digite a lista 2: |: [j,k,d,[[c]],b,j,a,j].
[[a,2],[b,2],[c,3],[d,2],[j,4]]
true .
?-


SWI-Prolog -- c:/Users/xand-laptop/Desktop/T1.pl
File Edit Settings Run Debug Help
Warning: c:/users/xand-laptop/desktop/t1.pl:7:
Singleton variables: [Lout]
Welcome to SWI-Prolog (threaded, 64 bits, version 7.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?- programa.
Digite a lista 1: [pato, [peixe,baleia], gato, [[[robo]]], [vaca,cavalo,[bezerro,cabrito]]].
Digite a lista 2: |: [gato, cabrito, robo, baleia, cavalo, zebra, elefante].
[[baleia,2],[gato,2],[robo,2],[cavalo,2],[cabrito,2]]
true .
?-

Código-fonte:

```
%T1
%Autores:
%Alexandre Regali Seleghim (RA 551473)
%Thiago Bretas de Souza (RA 551899)


programa:-
write('Digite a lista 1: '),
read(Lista1),
nl,
write('Digite a lista 2: '),
read(Lista2),
nl,
conta_atomos(Lista1,Lista2,Lout).




%L1, Lout
desparentize([], []).
desparentize([X|Y], [X|Z]) :- not(is_list(X)), desparentize(Y,Z), !.
desparentize([X|Y], Z) :- desparentize(X, X1), desparentize(Y, Y1), append(X1, Y1, Z).


%L1, L2, Lout
tira_nao_comuns([], _, []).
tira_nao_comuns([X1|Y1], L2, [X1|Z]) :- member(X1, L2), tira_nao_comuns(Y1, L2, Z).
tira_nao_comuns([_|Y1], L2, Z) :- tira_nao_comuns(Y1, L2, Z).


%Elemento, L1, N
conta(_, [], 0).
conta(X, [X | T], N) :- !, conta(X, T, N1), N is N1 + 1.
conta(X, [_ | T], N) :- conta(X, T, N).


%Elemento, L1, Lout
remove_elemento(_, [], []) :- !.
remove_elemento(X, [X|Y], Z) :- !, remove_elemento(X, Y, Z).
remove_elemento(X, [T|Y], Z) :- !, remove_elemento(X, Y, Y2), append([T], Y2, Z).


final([], Lout) :- length([], J), J =:= 0, write(Lout), !.
final([X|Y], Lout) :-
conta(X,[X|Y],N),
append(Lout,[[X,N]],Laux),
remove_elemento(X, [X|Y], Lout2),
final(Lout2, Laux).


conta_atomos([],[],[]).
conta_atomos(L1,L2,Lout):-
desparentize(L1, L11),
desparentize(L2, L22),
tira_nao_comuns(L11,L22, L111),
tira_nao_comuns(L22,L11, L222),
append(L111,L222, L3),
final(L3, Lout).
```