Trabalho 1 - Prolog

Disciplina

Paradigmas de Linguagem de Programação - 2018-1

Grupo

Carolina Junqueira Ferreira

Julio Brito

Exercício 1

Código

```
despar([],[]):- !.
despar([X1|Y],L):- X1 \== [], is_list(X1), despar(X1,R), despar(Y,R1),
append(R,R1,L), !.
despar([X1|Y],L):- not(atomic(X1)), despar(Y,L), !.
despar([X1|Y],[X1|Z]):- despar(Y,Z).

del_todas_ocorr(_,[],[]):- !.
del_todas_ocorr(X,[X|Y],Z):- del_todas_ocorr(X,Y,Z), !.
del_todas_ocorr(X,[X1|Y],[X1|Z]):- X \== X1, del_todas_ocorr(X,Y,Z).

conta_elem(_,[],0):- !.
conta_elem(X,[X|Y],N):- conta_elem(X,Y,N1), N is N1 + 1, !.
conta_elem(X,[X1|Y],N):- X \== X1, conta_elem(X,Y,N).

conta_todos([],[]):- !.
conta_todos([X|Y],[[X,Count]|R]):- conta_elem(X,Y,N), Count is N + 1,
del_todas_ocorr(X,Y,Z), conta_todos(Z,R).

conta(LIn,LOut):- despar(LIn,R), conta_todos(R,LOut).
```

Descrição dos Predicados

- **despar:** recebe uma lista e a desparentiza, eliminando também elementos que não são atômicos;
- **del_todas_ocorr:** remove todas as ocorrências de um elemento no primeiro nível de uma lista;
- **conta_elem:** conta as ocorrências de um elemento no primeiro nível de uma lista;
- **conta_todos:** conta as ocorrências dos elementos de uma lista dada e resulta numa segunda lista no formato pedido no exercício. Ex: [[a,2],[b,3],[[],2]].
- **conta:** recebe a lista LIn, a desparentiza e conta as ocorrências de seus elementos, resultando na LOut que possui o formato requerido no exercício Ex: [[a,2],[b,3],[[],2]].

Casos teste

- **Caso Exemplo:** [a,b,Z,x,4.6,[a,x],[],[5,z,x],[]]

```
?- conta([a,b,Z,x,4.6,[a,x],[],[5,z,x],[]], LOut).
LOut = [[a, 2], [b, 1], [x, 3], [4.6, 1], [[], 2], [5, 1], [z, 1]].
```

- **Caso 1:** [VAR,4.6,[],5,filho_de("maria","joao"),[a,z],[irmao("rui","joana"),VAR,[]],[5,z,x],[]]

```
?-
conta([VAR,4.6,[],5,filho_de("maria","joao"),[a,z],[irmao("rui","joana"),VAR,[]],[5,
z,x],[]],LOut).
LOut = [[4.6, 1], [[], 3], [5, 2], [a, 1], [z, 2], [x, 1]].
```

- **Caso 2**: [[],[],VAR,4.6,[a,[x]],[[[VAR,filho_de("maria","joao"),[]]]],[5,z,x],[]]

```
?-
conta([[],[],VAR,4.6,[a,[x]],[[[VAR,filho_de("maria","joao"),[]]]],[5,z,x],[]],LOut)
.
LOut = [[[], 4], [4.6, 1], [a, 1], [x, 2], [5, 1], [z, 1]].
```

Exercício 2

Código

```
tamanho([],0):- !.
tamanho([_|Cauda],N):- tamanho(Cauda,N1), N is N1 + 1.

monta_lista([],[]):- !.
monta_lista([[X|Y]|Cauda],[[Count,X]|Cauda2]):- tamanho(Y, N), Count is N + 1,
monta_lista(Cauda,Cauda2).

grup_elem([],[]):- !.
grup_elem([X,X|Cauda],Z):- grup_elem([[X,X]|Cauda],Z), !.
grup_elem([[X|Cauda1]|[X|Cauda2]],Z):- grup_elem([[X,X|Cauda1]|Cauda2],Z), !.
grup_elem([X|Cauda1],[[X]|Cauda2]):- not(is_list(X)), grup_elem(Cauda1,Cauda2), !.
grup_elem([X|Cauda1],[X|Cauda2]):- grup_elem(Cauda1,Cauda2).

conta_consec(LIn,Lout):- grup_elem(LIn,R), monta_lista(R,Lout).
```

Descrição dos Predicados

- **tamanho:** recebe uma lista e retorna o número de elementos dela;
- **grup_elem:** recebe lista e agrupa os itens repetidos consecutivamente. Por exemplo, se receber a

lista [a,a,a,a,b,c,c,a,a,d,e,e,e,e], retornará a lista [[a,a,a,a],[b],[c,c],[a,a],[d],[e,e,e,e]];
- **monta_lista:** constrói a lista no formato requerido no exercício. Ex: [[4,a],[1,b],[2,c],[2,a],[1,d],[4,e]];
- **conta_consec:** recebe uma lista LIn, agrupa os elementos repetidos consecutivamente através do "grup_elem", depois constrói a lista no formato desejado a partir do predicado "monta_lista".

Casos teste

- **Caso Exemplo:** [a,a,a,a,b,c,c,a,a,d,e,e,e,e]

```
?- conta_consec([a,a,a,a,b,c,c,a,a,d,e,e,e,e],Lout).
Lout = [[4, a], [1, b], [2, c], [2, a], [1, d], [4, e]].
```

- **Caso 1:** [a,a,[c,c],[c,c],[[c,c]],d,e,e,e,e]

```
?- conta_consec([a,a,[c,c],[c,c],[[c,c]],d,e,e,e,e],Lout).
Lout = [[2, a], [2, [c, c]], [1, [c, c]], [1, d], [4, e]].
```

- **Caso 2:** [filho_de(X,Y),filho_de(X,Y),a,a,a,e,b,c,c,filho_de(X,Y)]

```
?- conta_consec([filho_de(X,Y),filho_de(X,Y),a,a,a,e,b,c,c,filho_de(X,Y)],Lout).
Lout = [[2, filho_de(X, Y)], [3, a], [1, e], [1, b], [2, c], [1, filho_de(X, Y)]].
```

- **Caso 3:** [filho_de(X,Y):-teste(X,Y),filho_de(X,Y):-teste(X,Y),8.4,8.4,8.4,e,b,3,3,filho_de(X,Y)]

```
?-
conta_consec([filho_de(X,Y):-teste(X,Y),filho_de(X,Y):-teste(X,Y),8.4,8.4,8.4,e,b,3,
3,filho_de(X,Y)],Lout).
Lout = [[2, (filho_de(X, Y):-teste(X, Y))], [3, 8.4], [1, e], [1, b], [2, 3], [1,
filho_de(X, Y)]].
```

- **Caso 4:** [a,a,a,[],[],b,c,e,e,[[]],[[]],e]

```
?- conta_consec([a,a,a,[],[],b,c,e,e,[[]],[[]],e],Lout).
Lout = [[3, a], [2, []], [1, b], [1, c], [2, e], [2, [[]]], [1, e]].
```

- **Caso 5:** Não conta ocorrências de variáveis.

Exercício 3

Código

```
pega_elem([],[]):-!.
pega_elem([X1|Y],L):- pega_elem(Y,R), append(X1,R,L).
```

```
lista_rep(Elem,1,[Elem]):- !.
lista_rep(Elem,N,[Elem|Cauda]):- N1 is N - 1, lista_rep(Elem,N1,Cauda).

monta_lista([],[]):- !.
monta_lista([[Count,X]|Y],[Z|Cauda]):- lista_rep(X,Count,Z), monta_lista(Y,Cauda).

decode(LIn,Lout):- monta_lista(LIn,R), pega_elem(R,Lout).
```

Descrição de Predicados

- **pega_elem:** dada uma lista no formato [[a,a,a],[b],[[c,c],[c,c]] (elementos agrupados), retorna uma lista sem os parênteses do primeiro nível, ou seja, [a,a,a,b,[c,c],[c,c]];
- **lista_rep:** dada um elemento, número de repetições e lista de saída, resulta numa lista com X vezes o elemento passado. Por exemplo, lista_rep(a,4,R), R = [a,a,a,a];
- **monta_lista:** a partir no formato de lista de entrada do exercício, ex: [[4,a],[3,b],[1,[x,y]]], gera uma lista com sublistas no formato Ex: [[a,a,a,a],[b,b,b],[[x,y]]];
- **decode:** recebe lista LIn no formato de exemplo [[4,a],[3,b],[1,[x,y]]], então gera-se a decodificação da mesma para [a,a,a,a,b,b,b,b,[x,y]] a partir dos predicados "monta_lista" e "pega_elem", consecutivamente.

Casos teste

- **Caso Exemplo:** [[4,a],[1,b],[2,c],[2,a],[1,d],[4,e]]

```
?- decode([[4,a],[1,b],[2,c],[2,a],[1,d],[4,e]],Lout).
Lout = [a, a, a, a, b, c, c, a, a, d, e, e, e, e].
```

- **Caso 1:** [[3,[[]]],[1,b],[3,VAR],[2,3],[2,[]]]

```
?- decode([[3,[[]]],[1,b],[3,VAR],[2,3],[2,[]]],Lout).
Lout = [[[]], [[]], [[]], b, VAR, VAR, VAR, 3, 3, [], []].
```

- **Caso 2:** [[4,88],[3,filho_de(X,Y)],[2,[[a,b,c]]],[1,teste(X,H)],[2,"acre"]]

```
?- decode([[4,88],[3,filho_de(X,Y)],[2,[[a,b,c]]],[1,teste(X,H)],[2,"acre"]],Lout).
Lout = [88, 88, 88, 88, filho_de(X, Y), filho_de(X, Y), filho_de(X, Y), [[a, b,
c]], [[a, b, c]], teste(X, H), "acre", "acre"].
```