1. Uma lista é um ***termo*** que pode ter uma das seguintes formas:

a) `[]` (lista vazia)
b) um *par* `(cabeça, cauda)` (cabeça é o primeiro elemento e cauda é a lista com os restantes elementos) expresso de uma das formas:
1. `'.'(H,T)`
2. `[H|T]`

2. Defina os seguintes predicados para manuseamento de listas:

a) `membro/2`: dados um elemento X e uma lista L, `membro(X, L)` sucede se X ocorrer em L.

b) `prefixo/2`: dadas duas listas L1 e L2, `prefixo(L1, L2)` sucede se L1 for prefixo de L2. Comece por escrever (em Português) o que se entende por "ser prefixo de", continue com a especificação (sempre em Português) do que julgar ser um caso trivial e evidente, prossiga com um caso geral, entendido como sendo uma construção que diz como resolver um problema à custa doutro problema, da mesma natureza, mas mais simples. Finalmente, traduza para Prolog.

c) `sufixo/2`: dadas duas listas L1 e L2, `sufixo(L1, L2)` sucede se L1 é sufixo de L2. Faça como no ponto anterior.

d) `sublista/2`: dadas duas listas L1 e L2, `sublista(L1, L2)` sucede se L1 está contida em L2. Idem.

e) `concatena/3`: dadas listas L1, L2 e L3, `concatena(L1, L2, L3)` sucede se L3 é a concatenação de L1 com L2. Idem.

f) `inverte/2`: dadas duas listas L1 e L2, `inverte(L1, L2)` sucede se L2 é a inversão de L1. Idem.

f.1) Faça como nrev/2 visto nas teóricas.

f.2) Garanta que o predicado `inverte/2` tem recursividade terminal.

g) `tamanho/2`: `tamanho(L, T)` sucede se T é o comprimento da lista L.

g.2) Garanta que o predicado `tamanho/2` tem recursividade terminal.

Última alteração: quinta, 6 de outubro de 2022 às 09:57