

## Exercice à faire pour vendredi 20 octobre 2006 en début de TP (8h ou 13h45)

```

% membre(+E,+L) teste si E appartient à L
% en mode (-,+), donne tous les éléments de L
% aucun intérêt en mode (+,-)
membre(E,[E|_L]).
membre(E,[A|L]):- membre(E,L).

%test_en_serie
%'format' : prédicat predefini pour l'impression.
test_en_serie :-
    membre(X,[v1,v2,v3]),
    lancer(X),
    fail.
test_en_serie.

lancer(X) :-
    pred(X,R),
    !,
    format( "OK pour ~p. Le resultat est ~p.", [X,R]), nl.
lancer(X) :-
    format( "Echec pour ~p.", [X]), nl.

pred(X,V) :- c(X,V), V>10,!.
pred(X,V) :- c(X,V), V<5.

c(v1,12).    c(v2,7).    c(v3,4).

```

**Q-** Etant donné le programme ci-dessus, faire l'arbre de recherche de `test_en_serie`.

## Exercice - fait en TD-

Soit la grammaire suivante qui décrit des expressions préfixées d'arité variable telles que `(+ 2 5 6 )` ou `(+ 7 (* 3 4))`:

`E ::= <chiffre> | (+ L) | (* L)`

`L ::= E E | E L`

Le but de l'exercice est de construire en Prolog un analyseur du langage engendré par cette grammaire qui permette de calculer la valeur de l'expression analysée. La chaîne à analyser sera présentée sous forme d'une chaîne de caractères (cf `"(+ 2 5 6 )"`, c'est-à-dire la liste des codes Ascii).

Pour cela, on va procéder en deux temps:

**Q1-** *Grammaire dcg*: écrire un analyseur syntaxique pour ces expressions

**Q2-** *Ajout d'attributs*: compléter l'analyseur de façon à ce qu'il produise un terme évaluable représentant l'expression analysée.

Ainsi, sur les exemples, les termes produits seront `add(2, add(5,6))` et `add(7,mult(3,4))`.