TD de programmation fonctionnelle et logique
Corrigé du TD 9 : programmes Prolog et coupure

1. Écrivez, sans utiliser la primitive de division, un prédicat `division(A,B,Q,R)` où `Q` et `R` sont respectivement le quotient et le reste de la division de `A` par `B`.

```
division(A,B,Q,R) :-
   Bmoins1 is B-1, between(0,Bmoins1,R), between(0,A,Q), A =:= B*Q+R.
```

2. Écrivez un prédicat qui prend en entrée un élément et une liste et qui est vrai si l'élément appartient à la liste. Optimisez l'exécution de ce prédicat au moyen d'une coupure.

```
appartient(X,[X|_]) :- !.
appartient(X,[_|R]) :- appartient(X,R).
```

3. Écrivez un prédicat qui prend en entrée un élément et une liste et qui est vrai si l'élément appartient au moins deux fois à la liste.

```
appartientbis(X,[X|R]) :- !, appartient(X,R).
appartientbis(X,[_|R]) :- appartientbis(X,R).
```

4. Écrivez un prédicat `somme_cube(A,B,C)` qui est vrai si l'entier `C` est la somme des cubes des entiers strictement positifs `A` et `B`. Écrivez ce prédicat de telle sorte qu'il puisse, étant donné `C`, énumérer toutes les valeurs possibles de `A` et de `B`.

```
somme_cube(A,B,C) :-
   between(1,C,A), between(1,C,B), C is A*A*A+B*B*B.
```

5. Optimisez le prédicat de la question 4, sachant que l'on n'a plus besoin que d'une seule solution.

```
somme_cube(A,B,C) :-
   between(1,C,A), between(1,C,B), C is A*A*A+B*B*B, !.
```

6. Optimisez le prédicat de la question 4, sachant que l'on ne veut pas perdre de solutions.

```
somme_cube(A,B,C) :-
   s_cube(1,1,A,B,C).
s_cube(CA,_,_,_,C) :-
   2*CA*CA*CA  > C, !, fail.
s_cube(CA,CB,A,B,C) :-
   CA*CA*CA+CB*CB*CB > C, !, NCA is CA+1, s_cube(NCA,NCA,A,B,C).
s_cube(CA,CB,A,B,C) :-
   C =ˉ CA*CA*CA+CB*CB*CB, !, NCB is CB+1, s_cube(CA,NCB,A,B,C).
s_cube(A,B,A,B,C) :-  C is A*A*A+B*B*B.
s_cube(CA,_,A,B,C) :- NCA is CA+1, s_cube(NCA,NCA,A,B,C).
```

7. Écrivez un prédicat qui calcule le plus petit entier qui peut s'écrire, de deux manières différentes, comme la somme des cubes de deux entiers (ce nombre est inférieur à 10000).

```
plus_petit_double_cube(C) :-
   between(1,10000,C),somme_cube(A,_,C), somme_cube(D,_,C), A<D, !.
```

8. **Le motard généreux.** Un motard veut offrir une moto identique à chacune de ses petites amies. Le nombre représentant le coût de l'opération est, si l'on inverse tous les chiffres, celui qui représente le prix unitaire d'une moto. De mauvaises langues ont prétendu que le motard en question avait au moins deux petites amies, mais pas plus de 8. Le prix de la moto offerte se situant entre 10 000 et 99 999 francs, combien le motard a-t-il de petites amies et quel est le prix d'une moto ?

```
motard(F,M) :-
   between(2,8,F), between(1,9,A),between(0,9,B),between(0,9,C),
   between(0,9,D),between(0,9,E),
   M is A * 10000 + B * 1000 + C * 100 + D * 10 + E,
   F * M  =:= E * 10000 + D * 1000 + C * 100 + B * 10 + A.

F = 4
M = 21978 ;

F = 9
M = 10989
```