ENSEMBLES

On représente un ensemble (fini) d'objets par la liste de ses éléments. La liste vide représente donc l'ensemble vide. Une liste représente un ensemble si et seulement si elle est sans doublons.

L'opérateur `@` est l'opérateur de concaténation des listes. Il est de complexité linéaire par rapport à la première liste.

Un conseil : pensez récursivement !

1) Écrivez une fonction `unique`, qui supprime les doublons d'une liste :
`unique [1;2;1;0;3;2]` calcule la liste `[1;2;0;3]`

2) Écrivez les fonctions `union`, `intersection` , `difference` qui opèrent sur les ensembles. Estimez leurs complexités.

3) Écrivez une fonction `inclus` qui calcule le booléen indiquant si un ensemble est inclus dans un autre.

4) Écrivez une fonction `parties` qui calcule l'ensemble des parties d'un ensemble :
`parties [0;1;2]` calcule `[[]; [2]; [1]; [1; 2]; [0]; [0; 2]; [0; 1]; [0; 1; 2]]` ou tout autre liste à permutation prés.
Estimez sa complexité.

5) Écrivez une fonction `combinaisons` qui calcule les parties à k éléments d'un ensemble.

Plus difficiles :

6) Écrivez une fonction qui calcule l'ensemble des permutations d'un ensemble, c'est-à-dire si E est un ensemble de cardinal n, les $n!$ listes qui représentent l'ensemble.

7) On représente une application f d'un ensemble fini E dans un ensemble fini F par l'ensemble des couples $(x, f(x))$ où x parcourt E (c'est ce qu'on appelle le graphe de l'application).
Écrivez des fonctions qui calculent

a) l'ensemble des bijections entre deux ensembles de même cardinal ;

b) l'ensemble des injections entre deux ensemble ;

c) l'ensemble des surjections entre deux ensembles.