DS3 - Récursif

Récursif

Attention dans toute cette partie, vous n'avez pas le droit d'utiliser de références ni d'utiliser les modules `List` ou `Array`.
Les fonctions avec 🎄 en indication vous garantissent, si vous les faites toutes, une note de 15.

1. 🎄 Écrire une fonction `fact` calculant la factorielle de l'entier naturel passé en argument.
2. 🎄 Écrire une fonction `somme_liste` qui renvoie la somme des éléments de liste d'entiers passée en paramètre.
3. 🎄 Écrire une fonction `renverse` qui renvoie la liste passée en argument dans l'ordre inverse. **Attention** vous ne pourrez pas la programmer directement en récursif, il faut passer par une fonction auxiliaire qui prend un accumulateur. **Attention** vous aurez la moitié des points si votre fonction est en O(n^2).
4. 🎄 Ici et dans les deux questions qui suivents, on va considérer des listes du type `('a * 'b) list` qui permettent de stocker des associations entre des éléments du type `'a` et des éléments du type `'b`. C'est une manière de réaliser un dictionnaire. Ainsi, la liste `[ ("a", 42); ("toto", 12) ]` va associer la valeur 42 à la clé `"a"` et la valeur 12 à la clé `"toto"`. On appelle cette structure de donnée une liste d'associations.
   Écrire une fonction `ajoute_assoc : ('a * 'b) list -> 'a -> 'b -> ('a * 'b) list` qui prend une liste d'associations en entrée et a et b et rajoute une association entre a et b.
   Attention cette fonction est **très simple**.
5. 🎄 Écrire une fonction `cherche_assoc : ('a * 'b) list -> 'a -> 'b` qui prend une liste d'associations et une valeur a en entrée et renvoie la valeur b qui lui est associée ou produit une erreur avec l'instruction `failwith "invalide"`.
6. 🎄 Écrire une fonction `cherche_assoc_opt : ('a * 'b) list -> 'a -> 'b option` qui se comporte comme la fonction précédente hormis qu'elle utilise le type option pour renvoyer None en cas d'échec plutôt que de produire une erreur.
7. 🎄 Écrire une fonction `zip : 'a list -> 'b list -> ('a * 'b) list` qui prend deux listes **de même longueur** et renvoie la liste des couples d'un élément de la première liste associé avec l'élément correspondant de la seconde.
   Ainsi `zip [1;2;3] ['a';'b';'c']` doit renvoyer `[ (1,'a'); (2,'b'); (3,'c') ]`.
8. 🎄 Écrire une fonction `range : int -> int -> int list` tel que l'appel à `range a b` renvoie la liste d'entiers `[ a;a+1;...;b ]`.
9. *(Difficile)* Écrire une fonction `gen_tuples : 'a list -> int -> 'a list list` qui prend en entrée une liste `[a1;...;an]` et un entier p et renvoie tous les p-uplets de la liste sous forme de liste et dans l'ordre lexicographique correspondant à l'ordre de la liste. Ainsi pour `['a';'c';'b']` et 2 on va renvoyer `[ ['a';'a']; ['a';'c']; ['a';'b']; ['c';'a']; ['c';'c']; ['c';'b'];`