Manipulation des listes en Caml

L'objectif de ce TP est d'écrire les fonctions qui manipule la structure de donnée récursive List, implémentée dans le language Ocaml.

Pour chaque question, on prouvera la terminaison, la correction et la complexité de la solution proposée. On commencera systématiquement par déterminer le type de la fonction ou de l'objet à définir. On s'interdira bien entendu d'utiliser les fonctions existantes du module List, à l'exception des constructeurs et de la longueur.

1. Écrire une fonction `produit_cartesien` prenant en argument deux listes $l1$ et $l2$ et renvoyant une liste contenant tous les couples (a, b) où a est un élément de $l1$ et b un élément de $l2$.
2. Écrire une fonction `iteration_droite` prenant en argument une fonction f, une liste $[a_1; \ldots; a_n]$ et un élément b, et renvoyant $f\ a_1\ (f\ a_2\ (\ldots (f\ a_n\ b))\ldots)$.
   Cette fonction existe déjà sous le nom `List.fold_right`.
3. Retrouver la fonction `somme`, prenant une liste d'entiers et renvoyant la somme de ses éléments, en utilisant la fonction précédente. On pourra utiliser `(+)` l'opérateur somme vu comme une fonction `int -> int -> int`.
4. Écrire une fonction `iteration_gauche` prenant en argument une fonction f, un élément a et une liste $l = [b_1; \ldots; b_n]$, et renvoyant $f\ (\ldots (f\ (f\ a\ b_1)\ b_2)\ldots)\ b_n$.
   Cette fonction existe déjà sous le nom `List.fold_left`.
5. Retrouver `renverser` comme une application de la fonction précédente.
6. Écrire une fonction `tri_insertion` prenant en paramètre une liste d'entiers et la renvoyant triée par l'algorithme de tri par insertion qui consiste à trier la liste en partant de la fin et à rajouter les éléments au fur et à mesure dans cette liste triée.
7. Écrire une fonction `tribulle` prenant en paramètre une liste d'entiers et la renvoyant triée par l'algorithme du tri à bulles.
   On parcourt la liste et l'on compare les éléments consécutifs et lorsqu'ils ne sont pas dans le bon ordre on les échange. Après ce premier passage, le plus grand élément de la liste est à la fin. On recommence le nombre de fois nécessaire.
8. Écrire une fonction `tri_selection` prenant en paramètre une liste d'entiers et la renvoyant triée par extraction successives du minimum.
9. Écrire une fonction `trifusion` prenant en paramètre une liste d'entiers et la renvoyant triée par tri fusion, c'est-à-dire en la séparant en deux, en triant les deux sous-listes puis en les fusionnant.
10. Écrire une fonction qui prend en paramètre une liste et renvoie la liste de toutes ses sous-listes.
11. Écrire une fonction qui prend en paramètre une liste et renvoie la liste de toutes ses permutations.