Exercice 1 : Fonctions curryfiées et fonctions d'ordre supérieur sur les listes

1. (a) Écrire une fonction récursive[1] `concatenate : 'a list -> 'a list -> 'a list` qui prend en argument deux listes ℓ_1 et ℓ_2 et renvoie une liste composée des éléments de ℓ_1 suivis des éléments de ℓ_2 (dans le même ordre).
   (b) Quelle est l'ordre de grandeur de la complexité dans le pire cas de votre fonction ? Quel paramètre vous semble pertinent ici ?
2. (a) Écrire une fonction `rev_aux : 'a list -> 'a list -> 'a list` qui prend en argument deux listes ℓ_1 et ℓ_2 et renvoie une liste composée du renversé de ℓ_1, concaténé avec ℓ_2 (l'ordre de ℓ_2 ne change pas). Cette fonction doit être de complexité linéaire en la taille de ℓ_1 (vous le prouverez).
   (b) En déduire une fonction[2] `rev : 'a list -> 'a list` qui calcule le renversé d'une liste en temps linéaire.
3. Écrire une fonction[3] `map : ('a -> 'b) -> 'a list -> 'b list` telle que `map f [`x_0`; ...; `x_{n-1}`]` calcule `[f `x_0`; ...; f `x_{n-1}`]`.
4. Écrire une fonction[4] `for_all : ('a -> bool) -> 'a list -> bool` telle que `for_all p [`x_0`; `x_1`; ...; `x_{n-1}`]` renvoie `true` si et seulement si $\forall i \in [\![0; n-1]\!], \mathtt{p}(x_i)$.

Exercice 2 : Tableaux

1. (a) Écrire une fonction[5] `map : ('a -> 'b) -> 'a array -> 'b array` telle que `map f [|`x_0`; ...; `x_{n-1}`|]` calcule un **nouveau** tableau `[|f `x_0`; ...; f `x_{n-1}`|]`. Attention au cas du tableau vide quand vous créez le tableau image !
   (b) Écrire une fonction[6] `map_inplace : ('a -> 'a) -> 'a array -> unit` telle que `map_inplace f [|`x_0`; ...; `x_{n-1}`|]` **modifie** le tableau (sans rien renvoyer) pour que ses nouvelles valeurs soient `[|f `x_0`; ...; f `x_{n-1}`|]`.
   (c) Expliquer, à partir de la spécification (sans faire référence à votre code), les différences entre les types de ces deux fonctions.
2. Écrire une fonction[7] `array_of_list : 'a list -> 'a array` qui prend en argument une liste et renvoie un tableau ayant les mêmes éléments. Il vous faudra utiliser une fonction récursive auxiliaire qui n'a que des effets de bords pour remplir le tableau. Attention au cas de la liste vide !
3. Écrire une fonction[8] `list_of_array : 'a array -> 'a list` qui prend en argument un tableau et renvoie une liste ayant les mêmes éléments. Il vous faudra utiliser une fonction récursive auxiliaire prenant un argument de plus.

1. Elle correspond à l'opérateur binaire `@`.
2. Elle correspond à la fonction prédéfinie `List.rev`.
3. Elle correspond à la fonction prédéfinie `List.map`.
4. Elle correspond à la fonction prédéfinie `List.for_all`.
5. Elle correspond à la fonction prédéfinie `Array.map`.
6. Elle correspond à la fonction prédéfinie `Array.map_inplace`.
7. Elle correspond à la fonction prédéfinie `Array.of_list`.
8. Elle correspond à la fonction prédéfinie `Array.to_list`.