

Séance 2: LISTES CHAÎNÉES

L3 – Université Nice Sophia Antipolis

Exercice 1 (Fonctions élémentaires sur les listes, ★)

1. Écrivez une fonction récursive `produit : int list -> int` qui renvoie le produit d'une liste d'entiers. Optimisez votre fonction pour qu'elle renvoie 0 dès qu'un élément égal à 0 est vu dans la liste.
2. Écrivez une fonction `slice : 'a list -> int -> int -> 'a list` telle que `slice l a b` renvoie la sous-liste de `l` commençant à l'indice `a` inclus et terminant à l'indice `b` exclu. Par exemple, `slice [0;1;2;3;4;5] 2 4` renvoie `[2;3]`.
3. Redéfinissez la fonction `map : ('a -> 'b) -> 'a list -> 'b list` telle que `map f [x1;...;xn]` renvoie `[f x1;...;f xn]`.
4. Redéfinissez la fonction `filter ('a -> bool) -> 'a list -> 'a list` telle que `filter f l` renvoie la sous-liste des éléments `x` de `l` pour lesquels `f x` renvoie `true`.
5. Redéfinissez la fonction `assoc : 'a -> ('a * 'b) list -> 'b` telle que `assoc a [(a1,b1);...;(an,bn)]` renvoie le premier `bi` pour lequel `a=ai`. Si `a` n'est égal à aucun `ai`, une erreur est produite en appelant la fonction `invalid_arg`.

□

Exercice 2 (Retournement de liste, ★★)

1. Redéfinissez la fonction `rev : 'a list -> 'a list` qui renvoie l'image miroir d'une liste.
2. Quelle est la complexité de votre fonction ? On rappellera la complexité de la fonction `append` ?
3. Définissez la fonction `revappend l1 l2` qui renvoie la concaténation du miroir de `l1` et de `l2`.
Exemple : `revappend [1;2;3] [4;5]` s'évalue en `[3;2;1;4;5]`. Déduisez-en une autre définition possible de `rev` de meilleure complexité.

□

Exercice 3 (Décomposition en facteurs premiers, ★★)

Définissez une fonction `facteurs : int -> int list` qui renvoie la liste des facteurs premiers d'un nombre. Par exemple, `facteurs 84` renvoie `[2;2;3;7]` car $84 = 2 \times 2 \times 3 \times 7$. □

Exercice 4 (Tri insertion, ★★)

1. Écrivez une fonction `insert : 'a -> 'a list -> 'a list` qui insère un élément dans une liste triée en ordre croissant.
2. Déduisez-en une fonction `sort : 'a list -> 'a list` qui trie une liste en ordre croissant.

□

Exercice 5 (Ensembles et listes, ★★)

Convenons d'appeler ensemble une liste sans répétition dont l'ordre importe peu.

1. *Écrivez une fonction de conversion ensemble: 'a list -> 'a list qui supprime les répétitions. Par exemple, ensemble [1;2;3;1;1;4;2] renvoie [1;2;3;4].*
2. *Écrivez une fonction union: 'a list -> 'a list -> 'a list qui calcule l'union ensembliste de deux ensembles.*
3. *Écrivez une fonction cartésien: 'a list -> 'b list -> ('a * 'b) list qui calcule le produit cartésien de deux ensembles. Par exemple, cartésien [1;2;3] ['a';'b'] renvoie [(1,'a');(1,'b');(2,'a');(2,'b');(3,'a');(3,'b')].*

□