TD 1 - Remise à niveau

Dans le monde, il y a deux types de personnes, ceux qui ne connaissent pas le principe de récursion et ceux qui savent qu'il y a deux types de personnes, ceux qui ne connaissent pas le principe de récursion et ceux ...

Les listes

- Écrire une fonction `length : 'a list -> int` qui renvoi la longueur d'une liste.
- Écrire une fonction `mem : 'a -> 'a list -> bool` qui dit si un élément est dans une liste ou non.
- Écrire une fonction `rev : 'a list -> 'a list` qui fait le miroir d'une liste.
- Écrire une fonction `concat : 'a list -> 'a list -> 'a list` qui concatène deux listes.
- Écrire une fonction `iter : ('a -> 'b) -> 'a list -> 'b list` qui applique une fonction à tous les éléments d'une liste.
- Écrire une fonction `filter : ('a -> bool) -> 'a list -> 'a list` qui renvoi les éléments d'une liste qui vérifient un prédicat `p`.

Donner la complexité des fonctions écrites

Les tris

- Implémenter le tri bulle sur les tableaux sous la forme d'une fonction de type `'a vect -> unit`
- Implémenter le tri insertion sur les listes sous la forme d'une fonction de type `'a list -> 'a list`
- Implémenter le tri fusion sur les listes sous la forme d'une fonction de type `'a list -> 'a list`
- Implémenter le tri rapide sur les tableaux sous la forme d'une fonction de type `'a vect -> unit`

Pouvez vous donner la complexités des tries précédents en moyenne, meilleur cas, pire cas ?

Les structures

- Implémenter une structure de type LIFO (*Last In First Out*) avec les primitives :
  - `new_fifo : unit -> 'a lifo`
  - `is_empty : 'a lifo -> bool`
  - `push : 'a -> 'a lifo -> unit`
  - `pop : 'a lifo -> 'a`
- Même question avec une structure FIFO (*First In First Out*)

Testez vos fonctions.

Pour les plus rapides

Implémentez une structure d'ABR (Arbre binaire de rechercher) avec les fonctions suivante :

- `new_ABR : unit -> 'a abr`
- `insert : 'a -> 'a abr -> 'a abr`
- `remove : 'a -> 'a abr -> 'a abr`
- `abr_of_list : 'a list -> 'a abr`
- `list_of_abr : 'a abr -> 'a list`

En déduire une fonction de tri `sort : 'a list -> 'a list`

- Écrire une fonction de rééquilibrage d'arbre et l'appeler à l'insertion/suppression d'un élément de l'arbre.