T.P. 04.1
Les listes simplement chainées

■ ■ ■

Partie I : Echauffer les doigts

Exercice 1. (Bégaiement) Écrire une fonction récursive `begaie l` qui, étant donnée une liste `l = [a1;a2;...]` renvoie la liste `[a1;a1;a2;a2;...]`.
`val begaie : 'a list -> 'a list`

Exercice 2. Ecrire une fonction d'insertion dans une liste triée.

Exercice 3. (Flip)
- Programmer une fonction récursive qui met en premier le dernier élément d'une liste. C'est-à-dire `flip [1;2;3;4] = [4;1;2;3]`.
- On pourrait comme cela renverser une liste : renvoyer une liste ayant les mêmes éléments dans l'ordre contraire. En utilisant une liste auxilière (négligeament souvent appelé accumulateur), voyez vous une manière de renverser une liste en un seul parcours ?

Exercice 4. (Découpage de listes) Étant donnée une liste l et un pivot p, découper l selon p consiste à renvoyer deux listes éventuellements vides l_i et l_s telles que l_i contient les éléments de l strictement inférieurs à p et l_s contient les éléments de l supérieurs ou égaux à p, l'ordre des éléments de l étant conservé dans les listes l_i et l_s.

1. Écrire une fonction `decoupe l p` qui renvoie le couple de liste recherché.
`val decoupe l p : 'a list -> 'a -> ('a list * 'a list)`

2. Écrire une fonction `decoupe_strict l p` qui renvoie le triplet (l_i, l_s, np) tel que l_i (resp. l_s) est la sous-liste d'éléments de `l` strictement inférieurs (resp. strictement supérieurs) à `p` et np est le nombre d'occurrences de `p` dans `l`.
`val decoupe_strict : 'a list -> 'a -> ('a list * 'a list * int)`

3. En déduire une fonction `ieme l i` qui renvoie le $i$ème plus petit élément de la liste `l`, sans la classer.
`val ieme : 'a list -> int -> 'a`

Exercice 5. Expliquer le fonctionnement de la fonction suivante.

```
let mystere l =
  (* separe l = l1@l2 en (l1, l2) *)
  let rec avance l1 l2 =
    (* parcourt 2 fois plus vite l2 que l1 *)
    match l1, l2 with
    | _, [] -> [], l1
    | t1::q1, [ _ ] -> let a, l = avance q1 [] in (t1::a), l
    | t1::q1, _::_::q2 -> let a, l = avance q1 q2 in (h1::a), l
    | [], _ :: _ -> failwith ''impossible''
  in avance l l ;;
```

puis donner une autre implémentation d'une fonction qui couerait une liste en deux sous liste de taille au plus différente de 1.