TD n° 4 : Listes

EXERCICE 1 *Types de listes*

Donner le type des expressions suivantes, ainsi que ceux de `x`, `y` et `z` :

1. `[x]`
2. `x :: []`
3. `[] :: x`
4. `0 :: 1 :: x`
5. `0 :: 1 :: [x]`
6. `x :: y :: [z]`
7. `x :: [x]`
8. `x :: x`
9. `[x, y, z]`
10. `[x; [y; [z]]]`

EXERCICE 2 *Liste d'éléments de types différents*

On souhaite stocker des entiers *et* des flottants dans une même liste CAML. Est-ce possible ? Si oui, expliquer comment on peut procéder.

EXERCICE 3 *Produit des éléments d'une liste*

Écrire une fonction qui renvoie le produit des éléments d'une liste d'entiers.

EXERCICE 4 Déterminer le type et expliquer le comportement de la fonction suivante :

OCAML
```
let rec myst f liste =
  match liste with
  | [] -> []
  | tete :: queue when f tete -> tete :: (myst f queue)
  | tete :: queue -> myst f queue
;;
```

EXERCICE 5

La fonction prédéfinie `List.mem : 'a -> 'a list -> bool` teste l'appartenance d'un élément à une liste.

Un élève propose le programme suivant :

OCAML
```
let f x=  let rec y k= if List.mem k x
then y(k + 1)else k in y 0
;;
```

Réécrire ce programme de manière lisible. Ce programme est-il correct ? Si non, le corriger, donner son type, puis expliquer ce qu'il calcule et déterminer sa complexité.

EXERCICE 6 *Tri insertion*

On se propose de programmer le tri insertion (*insertion sort*) sur les listes. Le principe de ce tri est celui que vous utilisez pour trier un jeu de cartes en main : au fur et à mesure du tri, vous insérez un nouvel élément parmi ceux déjà triés.

1. Écrire une fonction `insere : 'a -> 'a list -> 'a list` prenant en argument un objet et une liste triée par ordre croissant et renvoyant la liste obtenue en insérant l'objet dans la liste en conservant son caractère trié. Par exemple, `insere 5 [2; 4; 7; 8; 9]` renvoie `[2; 4; 5; 7; 8; 9]`.
2. Écrire une fonction `tri_insertion : 'a list -> 'a list` triant la liste : lorsque c'est encore possible, il suffit d'insérer la tête dans la queue qui aura été triée récursivement.
3. Quelle est la complexité temporelle dans le meilleur et le pire des cas ? Préciser des listes pour lesquelles ces cas sont atteints.

EXERCICE 7 *Tri par sélection*

L'idée du tri par sélection (ou tri par extraction) est d'extraire le minimum d'une liste, de trier récursivement cette liste puis de placer ce minimum en tête.

Implémenter ce tri en CAML, en expliquant votre démarche, en donnant le type des fonctions utilisées et en analysant sa complexité dans le meilleur et dans le pire cas.

EXERCICE 8 *Les 2^n listes de n éléments de $\{0,1\}$*

Écrire une fonction `listes_binaires : int -> int list list` qui prend en argument un entier naturel n et qui retourne l'ensemble des suites des 2^n listes de n éléments pris dans l'ensemble $\{0,1\}$ classées par ordre lexicographique croissant. Le code des éventuelles fonctions intermédiaires doit être récursif.