TD n° 3 : 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.

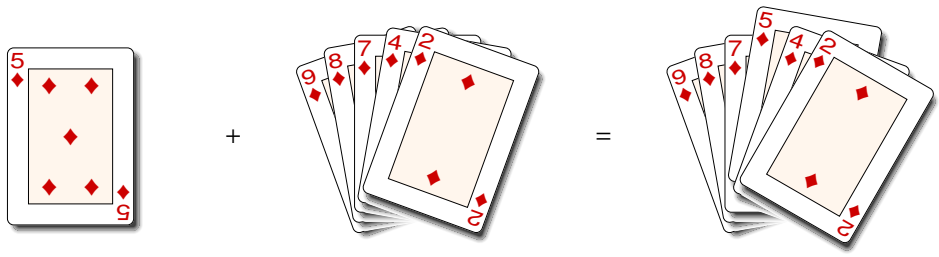
Exercice 4

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 *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.

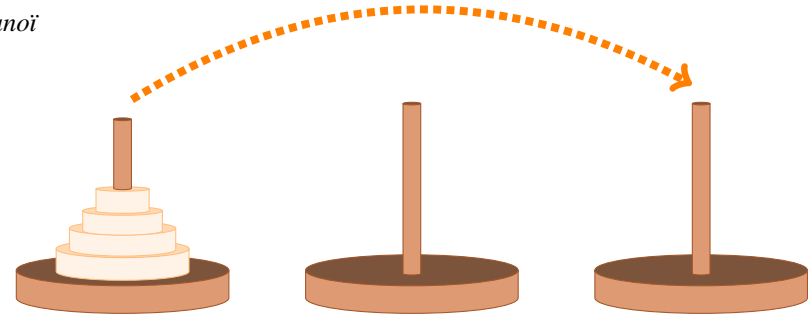
Exercice 6

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 7 *Tours de Hanoï*



Le jeu des **tours de Hanoï** a été inventé par le mathématicien Édouard Lucas au dix-neuvième siècle. Le principe est le suivant : il faut déplacer un pile de n disques d'un emplacement initial à un emplacement final en utilisant un emplacement intermédiaire, en respectant les règles :

- on ne peut déplacer que le disque en haut d'une pile,
- on ne peut empiler un disque que sur un disque plus grand.

1. Écrire une fonction `deplacement : int -> string -> string -> string` qui prend en argument un entier `n` et des chaînes de caractères `origine` et `destination` et renvoyant la chaîne de caractère `"Déplacer le disque {n} de {origine} à {destination}."` où `{n}`, `{origine}` et `{destination}` sont remplacés par leur valeur.
2. Écrire une fonction `hanoi_liste : int -> string -> string -> string -> string list -> string list` qui prend en argument :
   - un nombre entier `n` de disques à déplacer,
   - des chaînes de caractères `debut`, `fin` et `intermediaire` décrivant les piquets de départ et d'arrivée pour le déplacement des `n` disques, ainsi que le troisième piquet,
   - une liste de déplacements (chaînes de caractères) `liste_depl` déjà effectués

   et qui :
   - renvoie la liste de déplacements s'il n'y en a pas d'autre à faire,
   - résout sinon le problème de déplacements des n disques, à l'aide de deux déplacements de $n-1$ disques (appels récursifs) et du déplacement d'un disque (appel à `deplacement`) et renvoie la liste de tous les déplacements effectués.
3. Écrire une fonction `hanoi : int -> string list` renvoyant la liste de tous les déplacements à effectuer pour résoudre le problème avec un nombre donné de disque.

   OCaml
   ```
   hanoi 3;;
   - : string list =
   ["Deplacer le disque 1 de gauche a droite.";
    "Deplacer le disque 2 de gauche a milieu.";
    "Deplacer le disque 1 de droite a milieu.";
    "Deplacer le disque 3 de gauche a droite.";
    "Deplacer le disque 1 de milieu a gauche.";
    "Deplacer le disque 2 de milieu a droite.";
    "Deplacer le disque 1 de gauche a droite."]
   ```

4. Combien de déplacements sont nécessaires pour déplacer n disques ?