Colle 06

Cette sixième colle vous fera reconnaître du code OCaml, et écrire quelques fonctions simples en OCaml sur des listes ou des tableaux.

Une petite fonction sur une liste

1. Écrivez en OCaml une fonction `minimum_liste` qui prend en entrée une liste (de type `'a list`) et renvoie son élément minimum. La fonction devra être récursive et ne pas utiliser `List.nth` mais un parcours récursif de la liste avec du *pattern matching*. On rappelle que la fonction binaire `min x y` renvoie le minimum entre deux valeurs `x` et `y` ;
2. Testez la sur plusieurs listes. Que va-t-il se passer sur une liste vide ? Expliquez.

Calcul vectoriel sur des tableaux

On cherche à écrire des fonctions qui calculent sur des vecteurs de $\overrightarrow{x} \in \mathbb{R}^n$, c'est-à-dire des tableaux de flottants en OCaml de taille n.

1. Donnez à l'écrit le type d'un vecteur en OCaml ;
2. Écrivez une fonction `norme` qui prenne un vecteur et renvoie sa norme L^2 définie par $|\overrightarrow{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$. On rappelle que les principales fonctions mathématiques sont disponibles sans avoir besoin de les importer, grâce au module `Stdlib` qui est entièrement importé par défaut (sa documentation est en ligne sur `https://ocaml.org/api/Stdlib.html` si besoin) ;
3. Écrivez une fonction `somme` qui prend deux vecteurs ayant la même longueur et renvoie leur somme. On pourra consulter la documentation du module `Array` sur `https://ocaml.org/api/Array.html` pour se rafraîchir les idées quant aux fonctions `Array.init` ou `Array.make` dont voici les signatures :

```
# Array.make;;
- : int -> 'a -> 'a array = <fun>
# Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
```

4. De même, écrivez une fonction `difference` qui prend deux vecteurs ayant la même longueur et renvoie leur différence.
5. Enfin, écrivez une fonction `produit_exterieur` qui prend un vecteur (non vide) et un scalaire et renvoie le produit extérieur du vecteur par ce scalaire, c'est-à-dire $\lambda.\overrightarrow{x} = [\lambda x_1, \ldots, \lambda x_n]$.

Lecture de code et analyse de types de fonctions

Déterminer le type des fonctions suivantes. Certaines définitions sont équivalentes et ne diffèrent que par la syntaxe utilisée. Préciser quelles sont les définitions équivalentes.

Il ne faut **pas** utiliser le terminal OCaml pour répondre à ces questions.


```
let f01 x y = x
let f02 x = function y -> y
let f03 x y = x y
let f04 x y = y x
let f05 x y = x y y
let f06 x y = x (x y)
let f07 = function x -> function y -> y (x y)
let f08 x y = (x y) y
let f09 x y = x (y x)
let f10 x y = y (x y)
```

(Exercice emprunté à Nicolas Pécheux de la MP2I du lycée Carnot à Dijon.)

