TD n° 2 : Récursivité

EXERCICE 1 La fonction suivante n'est pas récursive terminale

```
let rec itere n f x =
  match n with
  | 0 -> x
  | n -> f (itere (n - 1) f x)
;;
```

Comment rendre cette fonction récursive terminale ?

EXERCICE 2 (*Concaténation récursive terminale*)

1. Rappeler comment écrire en CAML une fonction `concat` qui concatène deux listes (c'est-à-dire comment implémenter `@`). Quelle est sa complexité ? Cette fonction est-elle récursive terminale ?
2. Écrire une fonction `rev_append` de type `'a list -> 'a list -> 'a list` telle que `rev_append l1 l2` renvoie le renversement de `l1` concaténé avec `l2`. Cette fonction est-elle récursive terminale ? Quelle est sa complexité ?
3. En déduire la fonction `rev : 'a list -> 'a list`
4. En déduire une fonction `append : 'a list -> 'a list -> 'a list` qui concatène deux listes. Quelle est sa complexité ? Est-elle récursive terminale ?

EXERCICE 3 La fonction `sigma` vue en cours n'est pas récursive terminale :

```
let rec sigma n =
  match n with
  | 0 -> 0
  | n -> n + sigma (n - 1)
;;
```

En utilisant une fonction auxiliaire et un accumulateur, écrire une version récursive terminale de cette fonction somme.

EXERCICE 4 Réécrire la fonction `length : 'a list -> int` de façon à ce qu'elle soit récursive terminale [1].

1. C'est bien le cas de la fonction `list_length` fournie par CAML.

EXERCICE 5 (*Ordre de Sarkovski*) L'ordre de Sarkovski sur $\mathbb{N}^*$ est défini par :

$$\begin{aligned} 1 \prec 2 \prec 2^2 \prec 2^3 \prec \dots \\ \prec 2^3 \times 5 \prec 2^3 \times 3 \prec \dots \\ \prec 2^2 \times 5 \prec 2^2 \times 3 \prec \dots \\ \prec 2 \times 5 \prec 2 \times 3 \prec \dots \\ \prec 9 \prec 7 \prec 5 \prec 3 \end{aligned}$$

1. Comparer 1988, 1989 et 1990 pour cet ordre.
2. Est-ce un ordre bien fondé sur $\mathbb{N}^*$?

EXERCICE 6 (*Caractéristation équivalente d'un ensemble bien fondé*) Montrer qu'un ensemble ordonné $(E, \preccurlyeq)$ est bien fondé si et seulement s'il n'existe pas de suite strictement décroissante dans E.

EXERCICE 7 (*Fonction 91 de McCarthy*) Déterminer ce que calcule la fonction suivante et le démontrer :

```
let rec f_91 n =
  if n > 100 then
    n - 10
  else
    f_91 (f_91 (n + 11))
;;
```

Remarquons qu'il n'était pas évident *a priori* qu'une telle fonction renvoie bien un résultat.

EXERCICE 8 (*Fonction d'Ackermann*) La fonction d'Ackermann est définie sur $\mathbb{N} \times \mathbb{N}$ par :

```
let rec ackermann n p =
  match n, p with
  | 0, p -> p + 1
  | n, 0 -> ackermann (n - 1) 1
  | n, p -> ackermann (n - 1) (ackermann n (p - 1)) ;;
```

Est-elle récursive terminale ? Démontrer la terminaison de cette fonction pour tout couple d'arguments entiers naturels.