TD n° 4 — corrigé

EXERCICE 1 Il suffit de remarquer que $f\left(f^{n-1}(x)\right) = f^n(x) = f^{n-1}(f(x))$.

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

EXERCICE 2 (*Concaténation récursive terminale*)

1. La récursivité n'est pas terminale.

   OCAML
   ```
   let rec concat liste1 liste2 =
     match liste1 with
     | [] -> liste2
     | tete :: queue -> tete :: (concat queue liste2)
   ```

   Si n_1 et n_2 sont les tailles des deux listes, $C(0, n_2) = 0$ et si $n_1 > 0$,

   $$C(n_1, n_2) = C(n_1 - 1, n_2) + \Theta(1).$$

   Donc $C(n_1, n_2) = \Theta(n_1)$ (ne dépend pas de n_2).

2. La récursivité est terminale.

   OCAML
   ```
   let rec rev_append liste1 liste2 =
     match liste1 with
     | [] -> liste2
     | tete :: queue -> rev_append queue (tete :: liste2)
   ```

   Si n_1 et n_2 sont les tailles des deux listes, $C(0, n_2) = 0$ et si $n_1 > 0$,

   $$C(n_1, n_2) = C(n_1 - 1, n_2 + 1) + \Theta(1).$$

   Donc

   $$\begin{aligned} C(n_1, n_2) &= C(n_1 - 1, n_2 + 1) + \Theta(1) \\ &= C(n_1 - 2, n_2 + 2) + \Theta(1) + \Theta(1) \\ &\vdots \\ &= C(0, n_1 + n_2) + \sum_{k=0}^{n_1} \Theta(1) \\ &= \Theta(n_1) \end{aligned}$$

   (ne dépend pas de n_2).

3. On a immédiatement :

   OCAML
   ```
   let rev liste = rev_append liste []
   ```

4. Cette fonction n'est pas récursive mais ne fait intervenir que des fonctions récursives terminales.

   OCAML
   ```
   let append liste1 liste2 =
     rev_append (rev liste1) liste2
   ```

   Complexité : $C(n_1, n_2) = C_{\texttt{rev\_append}}(n_1, n_2) + C_{\texttt{rev}}(n_1) = \Theta(n_1)$.

EXERCICE 3

OCAML
```
let sigma n =
  (* Invariant de sigma_aux : somme(k, k = 0 .. n) + accu *)
  let rec sigma_aux n accu =
    match n with
    | 0 -> accu
    | _ -> sigma_aux (n - 1) (accu + n)
  in
  sigma_aux n 0
```

EXERCICE 4

OCAML
```
let length liste =
  (* Invariant de length_aux : longueur(liste) + compteur *)
  let rec length_aux liste compteur =
    match liste with
    | [] -> compteur
    | _ :: queue -> length_aux queue (succ compteur)
  in
  length_aux liste 0
```

EXERCICE 5

1. Comme l'addition et la multiplication sont associatives, on peut, pour les trois fonctions, utiliser aussi bien `List.fold_right` que `List.fold_left` (préférable car récursive terminale).

OCAML
```
let length = List.fold_left (fun cpt _ -> succ cpt) 0

let length liste =
  List.fold_right (fun _ cpt -> succ cpt) liste 0

let somme = List.fold_left (fun x y -> x + y) 0

let produit liste =
  List.fold_right (fun x y -> x * y) liste 1
```

2. On implémente `fold_left` avec une récursivité terminale.

OCAML
```
let rec fold_left f elt liste =
  match liste with
  | [] -> elt
  | tete :: queue -> fold_left f (f tete) queue

let rec fold_right f liste elt =
  match liste with
  | [] -> elt
  | tete :: queue -> f tete (fold_right f queue elt)
```

3. La fonction `myst1 : 'a -> 'a list -> 'a list` ajoute un élément en queue de liste. La fonction `myst2 : 'a list -> 'a list -> 'a list` est la concaténation (`@`) et la fonction `myst3 : 'a list -> 'a` renvoie le minimum d'une liste.
4. `let map f lst = fold_right (fun x l -> (f x) :: l) lst []`

EXERCICE 6 (*Fonction 91 de McCarthy*)

- Si $n > 100$, $f_{91}(n) = n - 10$.
- Si $n = 100 - k \in [\![90, 100]\!]$ avec $k \in [\![0, 10]\!]$, on montre par récurrence simple finie sur k que $f_{91}(n) = 91$.
  - C'est vrai pour $k = 0$ car
  $$f_{91}(100) = f_{91}(f_{91}(111)) = f_{91}(101) = 91,$$
  - Si c'est vrai pour un $k \in [\![0, 9]\!]$, alors
  $$f_{91}(100 - (k+1)) = f_{91}(99 - k) = f_{91}(f_{91}(110 - k))$$
  avec $110 - k > 100$ donc
  $$f_{91}(100 - (k+1)) = f_{91}(100 - k) = 91$$
  par hypothèse de récurrence.

☞ **Remarque 1**

On peut aussi se passer de récurrence : si $n \in [\![91, 100]\!]$,
$$f_{91}(n) = f_{91}(f_{91}(n+11)) = f(n+1)$$
car $n + 11 > 100$. Ainsi, pour un tel n, $f_{91}(n) = f_{91}(101) = 91$.
(En fait la récurrence est implicite. Où est-elle ?)

- Puis, si $n = 100 - k \leqslant 100$ avec $k \in \mathbb{N}$, on montre par récurrence sur k que $f_{91}(n) = 91$.
  - C'est vrai pour $k \in [\![0, 10]\!]$ d'après le point précédent et *toutes ces initialisations sont nécessaires*.
  - Si pour un $k \geqslant 11$, c'est vrai pour $k - 11$, alors
  $$f_{91}(100-k) = f_{91}(f_{91}(111-k)) = f_{91}(f_{91}(100-(k-11))) = f_{91}(91) = 91$$
  par hypothèse de récurrence puis par le cas $k = 9$.