Exercices sur la récursivité

MP2I Lycée Pierre de Fermat

Exercice 1.

Récursivité terminale

```
(** Q1 **)

(* somme des éléments de l, plus r *)
let rec somme_add (l: int list) (r: int) : int =
  match l with
  | [] -> r
  | x :: q -> somme_add q (x + r)

(* somme des éléments de l *)
let somme (l: int list): int =
  somme_add l 0

(** Q2 **)
(* on utilise la fonction max: 'a -> 'a -> 'a
   qui calcule le maximum de deux éléments *)

(* max des éléments de l. Précondition: l non vide *)
let max_liste (l: 'a list) : 'a =
  (* max de m et des éléments de ll *)
  let rec max_elem (ll: 'a list) (m: 'a) : 'a =
    match ll with
    | [] -> m
    | x :: q -> max_elem q (max x m)
  in
  match l with
  | [] -> failwith "Max d'une liste vide"
  | x :: q -> max_elem q x

(** Q3 **)

(* Liste des éléments de l qui vérifient p *)
let filter (p: 'a -> bool) (l: 'a list): 'a list =
  (* Liste des éléments de ll qui vérifient p, renversée,
     concaténée à accu *)
  let rec rev_filter_concat (l: 'a list) (accu: 'a list) : 'a list =
    match l with
    | [] -> accu
    | x :: q ->
      if p x then rev_filter_concat q (x :: accu)
      else       rev_filter_concat q accu
  in
  List.rev (rev_filter_concat l [])

(** Q4 **)
```

```
45
46 (* fusion triée de l1 et l2, renversée, concaténée à accu
47  * Précondition: l1 et l2 sont triées *)
48 let rec rev_fusion_concat (l1: 'a list) (l2: 'a list) (accu: 'a list): 'a list =
49   match (l1, l2) with
50     | [], _ -> List.rev l2 @ accu
51     | _, [] -> List.rev l1 @ accu
52     | x1 :: q1, x2 :: q2 ->
53       if x1 <= x2 then
54         rev_fusion_concat q1 l2 (x1 :: accu)
55       else
56         rev_fusion_concat l1 q2 (x2 :: accu)
57
58
59 (* fusion triée de l1 et l2
60  * Précondition: l1 et l2 sont triées *)
61 let fusion (l1: 'a list) (l2: 'a list) : 'a list =
62   List.rev (rev_fusion_concat l1 l2 [])
63
64 (** Q5 **)
65
66 (* couple (l1, l2) formant une partition de l, avec
67  * - l1 liste des éléments inférieurs ou égaux à p
68  * - l2 liste des éléments supérieurs strict à p *)
69 let partition (p: 'a) (l: 'a list) : 'a list * 'a list =
70   (* partitionne ll selon p et rajoute les éléments
71      inférieurs et supérieurs stricts dans les listes
72      respectives l1 et l2. Renvoie le couple des listes
73      obtenues. *)
74   let rec partition_add (ll: 'a list) (l1: 'a list) (l2: 'a list) : 'a list * 'a list =
75     match ll with
76     | [] -> (l1, l2)
77     | x :: q ->
78       if x <= p then partition_add q (x :: l1) l2
79       else          partition_add q l1 (x :: l2)
80   in
81   partition_add l [] []
```

Exercice 2.

Correction du tri fusion

Q1. On propose la propriété suivante :

$P(n)$: " Pour toutes listes L_1, L_2 triées telles que $|L_1| + |L_2| = n$, **fusion**(L_1, L_2) est triée et contient les éléments de L_1 et L_2."

Notons qu'on ne peut **pas** raisonner par récurrence juste sur $|L_1|$ ou sur $|L_2|$ car aucune de ces deux quantités n'est systématiquement strictement décroissante au cours des appels récursifs, on ne pourrait donc pas appliquer l'hypothèse de récurrence.

Q2. Montrons $\forall n, P(n)$ par récurrence :

— Initialisation ($n = 0$) : Soient L_1, L_2 triées telles que $|L_1| + |L_2| = 0$. Alors, $L_1 = [] = L_2$. Alors :

$$\textbf{fusion}(L_1, L_2) \quad = \quad []$$

Cette liste est triée, et contient bien les éléments de L_1 et L_2, qui sont vides. D'où $P(0)$.

— Hérédité : Soit $n \in \mathbb{N}$ tel que $P(n)$. Soient L_1, L_2 triées telles que $|L_1| + |L_2| = n+1$. Trois cas se présentent :

— Si L_1 est vide, alors $\textbf{fusion}(L_1, L_2) = L_2$. C'est bien une liste triée par hypothèse, et elle contient trivialement les éléments de L_1 (qui est vide) et de L_2.

— Si L_2 est vide, c'est analogue.

— Si aucune des deux n'est vide, alors $L_1 = x_1 :: Q_1$ et $L_2 = x_2 :: Q_2$ avec x_1, x_2 des éléments et Q_1, Q_2 des listes. Sans perte de généralité, supposons $x_1 < x_2$, l'autre cas étant identique. Alors :

$$\textbf{fusion}(L_1, L_2) \quad = \quad x_1 :: \textbf{fusion}(Q_1, L_2)$$

De plus, $|Q_1| + |L_2| = n$, donc, par hypothèse de récurrence, on a $\textbf{fusion}(Q_1, L_2)$ qui est triée et contient les éléments de Q_1 et de L_2. Alors, il est clair que $\textbf{fusion}(L_1, L_2)$ contient les éléments de $L_1 = x_1 :: Q_1$ et de L_2. De plus, $\textbf{fusion}(Q_1, L_2)$ est triée, et contient uniquement des éléments supérieurs à x_1. En effet, $x_1 < x_2 = \min(L_2)$ car L_2 est triée. Ainsi, $\textbf{fusion}(L_1, L_2)$ est aussi triée.

Dans tous les cas, $P(n+1)$ est vérifiée, d'où l'hérédité.

D'où la correction de la fonction **fusion**.