info421 : Programmation fonctionnelle
TD 6 : exceptions

Responsable : Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
email : Pierre.Hyvernat@univ-savoie.fr
http://lama.univ-savoie.fr/~hyvernat/

Exercice 1 : exceptions : listes

Question 1. Programmez la fonction `last : 'a list -> 'a` qui renvoie le dernier élément d'une liste.

Question 2. Programmez une fonction `element : int -> 'a list -> 'a` qui renvoie le n-ème élément d'une liste.

Question 3. La fonction `supprime : 'a -> 'a list -> 'a list` peut avoir deux comportements lorsqu'on essaie de supprimer un élément qui n'existe pas :
- on renvoie la liste originale,
- on provoque une exception.

Écrivez les deux variantes, et comparez les.

Question 4. Écrivez et comparez les deux variantes de `supprime` quand on sait que les listes sont triées.

Exercice 2 : exceptions : `fold`

Les fonctions `fold_right` et `fold_left` sont bien pratiques, mais elles parcourent toujours la liste en entier. Ceci vient du fait que dans la définition

```
let rec fold_right f l o =
  match l with
    [] -> o
  | a::l -> f a (fold_right f l o)
```

Caml doit toujours calculer "`fold_right f l o`" avant de pouvoir appeler la fonction `f`.

Question 1. En utilisant `List.fold_left`, programmez une fonction `produit` qui calcule le produit des éléments d'une liste d'entiers.

Question 2. Idem, mais pour une fonction `produit7` qui calcule le produit modulo 7. (L'opérateur "`mod`" de Caml permet de calculer le modulo.)

Question 3. À votre avis, que se passe-t'il si on essaie de calculer `produit7 l` quand `l` est une grande liste d'entiers aléatoires entre 0 et 6 ?

Réécrivez `produit7` *toujours avec* `List.fold_left`, mais en utilisant une exception pour améliorer le calcul.

Exercice 3 : exceptions : autres exemples

Question 1. Modifiez la fonction `suppr : 'a -> 'a arbre -> 'a arbre` qui qui supprime la première occurrence d'un élément donné dans un arbre binaire de recherche pour qu'elle renvoie l'exception `Not_found` lorsque l'élément n'est pas présent dans l'arbre.

Peut-on simuler l'ancien comportement avec le nouveau ?

Peut-on simuler le nouveau comportement avec l'ancien ?

Question 2. Quel est l'avantage de la nouvelle version de `supprime` ?

Question 3. La fonction `open_in "..."` permet d'ouvrir un fichier, et la fonction `input_line f` permet de récupérer la prochaine ligne dans le fichier `f` :

```
let f = open_in "fichier.txt" in
let l1 = input_line f in
let l2 = input_line f in
...
```

Lorsqu'il n'y a plus de ligne dans le fichier, la fonction `input_line` lève l'exception `End_of_file`. Écrivez une fonction qui ouvre puis récupère toutes les lignes d'un fichier dans une liste.

Question 4. Tartempion a écrit

```
let tete l = match l with
    [] -> raise Not_found
  | a::_ -> a

let queue l = match l with
    [] -> []
  | a::l -> l

let rec map_exn f l =
  try
    let a, l = tete l, queue l in
      (f a)::map_exn f l
  with
    Not_found -> []
```

Que fait cette fonction :
- lorsque `f` ne lève pas d'exception ?
- lorsque `f` lève une exception ?