info401 : Programmation fonctionnelle
Contrôle des connaissances – 3

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/
wiki : http://www.lama.univ-savoie.fr/wiki

Tout comme pour les TP, n'oubliez pas de commentez votre code pour préciser les points importants.

N'hésitez pas à utiliser les fonctions de la librairie `list` :
- `map : ('a -> 'b) -> 'a list -> 'b list`,
- `fold_right : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b`,
- `fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a`,
- `...`

Un point est réservé pour la présentation.

Partie 1 : petits exercices

(2) *Question 1.* Programmez la fonction `applique : ('a->'b) list -> 'a list -> 'b list` qui applique toutes les fonctions d'une liste à tous les arguments d'une autre liste. Par exemple
```
# applique [ fun x -> x-1 ; fun x -> x*x ] [1;2;3] ;;
- : int list = [ 0; 1; 2; 1; 4; 9 ]
```
Remarque : l'ordre de la liste résultat n'est pas important.

(2) *Question 2.* La fonction `flatten : 'a list list -> 'a list` permet de concaténer toutes les listes présentes dans une liste. Par exemple :
```
# List.flatten [ [1;2] ; [3;4] ; [3;4] ; [] ; [5] ] ;;
- : int list = [1; 2; 3; 4; 3; 4; 5]
```
Cette fonction n'est pas récursive terminale.

Écrivez une fonction `flatten_tr : 'a list list -> 'a list` en faisant en sorte qu'elle soit récursive terminale.

Quels sont les avantages / inconvénients de `flatten` par rapport à `flatten_tr` ?

(2) *Question 3.* Si `f` est une fonction de type `float -> float`, on souhaite calculer une approximation de sa `dérivée`. En mathématique, la définition est

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

Comme on ne sait pas calculer la limite exacte on rajoute un paramètre `e` : "`derive f e`" devra donc calculer l'approximation de la dérivée de `f` avec $\varepsilon =$ `e`.

Programmez la fonction `derive` et donnez son type.

(2) *Question 4.* On suppose que la fonction `insere : int -> int list -> int list` insère correctement un entier dans une liste triée. Que font les fonctions
```
let mystere1 l = List.fold_right (fun a r -> insere a r) l [];;
let mystere2 l = List.fold_left (fun r a -> insere a r) [] l;;
```
Justifiez attentivement votre réponse.

Partie 2 : problème : matrices

Le but de cette partie est d'écrire le début d'une bibliothèque (non optimisée) pour les manipulation de *matrices*, c'est à dire de tableaux bidimensionnels de nombres.

Chaque matrice sera vu comme une liste de lignes numérotées, et chaque ligne sera vu comme une liste de "cases" numérotées. Par exemple, la matrice

$$\begin{pmatrix} 2.5 & 0 \\ 0 & 1.2 \end{pmatrix}$$

sera représentée par

```
[ (0, [ (0, 2.5) ; (1, 0.0)]) ;     (* ligne 0 *)
  (1, [ (0, 0.0) ; (1, 1.2)]) ]     (* ligne 1 *)
```

Important :

- on numérote les lignes et les cases à partir de 0,
- les lignes et les cases peuvent être dans le désordre,
- si une case est omise, on considère que sa valeur est 0.0,
- si une ligne est omise, on considère que cette ligne ne contient que des 0.0.

Par exemple, la matrice précédente pourrait également être représentée par

```
[ (1, [ (1, 1.2)]) ;
  (0, [ (1, 0.0) ; (0, 2.5)])]
```

(1) *Question 1.* Donnez la définition du type `'a matrice` pour les matrices contenant des cases de type `'a`.

(2) *Question 2.* Écrivez deux fonctions

- `hauteur : 'a matrice -> int`
- et `largeur : 'a matrice -> int`

qui calculent respectivement la hauteur et la largeur d'une matrice.

Remarque : on s'intéresse plus précisément à la hauteur / largeur de la représentation de la matrice en ne regardant que les cases / lignes présentes dans la représentation.

(2) *Question 3.* Écrivez une fonction `case : int -> int -> 'a matrice -> 'a` qui permet de trouver la valeur d'une case d'une matrice.

Vous pouvez pour ceci vous aider de la fonction `List.assoc : 'a -> ('a*'b) list -> 'b`.

(2) *Question 4.* Écrivez une fonction `transpose : 'a matrice -> 'a matrice` qui calcule la transposée d'une matrice, c'est à dire qui renverse les lignes et les colonnes d'une matrice.

N'hésitez pas à utiliser des fonctions auxiliaires...

(1) *Question 5.* Pour rassembler les fonctions précédentes sur les matrices (et d'autres), on souhaite faire un module.

Donnez une signature en Caml qui pourrait être la signature d'un tels module, et qui rende le type des matrices abstraits.

(1) *Question 6.* Une opération habituelle sur les matrice est la multiplications. Pour pouvoir la faire, il faut savoir comment additionner et multiplier les cases entre elles.

Comment pourrait-on procéder pour faire un foncteur `Matrice` qui construise un module de signature appropriée.

Sans le programmer explicitement, donnez des détails sur la procédure à suivre et le choix de paramètre(s) de ce foncteur.

(1) *Question 7.* Comment pourrait-on améliorer la complexité des calculs sur les matrices ?