

Interrogation écrite 4

INF 201 — IMA4 — 27/04/2023 — 20 minutes

Problème. (/21) Dans cet exercice il est **interdit** d'écrire des fonctions récursives, il faudra utiliser à la place les schémas d'ordre supérieur du module `List`. Vous êtes autorisés à appeler les fonctions en omettant le `List.`, par exemple `hd` au lieu de `List.hd`.

En mathématiques, on nomme fraction continue une expression de la forme:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$$

où $a_0 \in \mathbb{Z}$ et $a_{i,i>0} \in \mathbb{N}$. On peut avoir un nombre fini ou infini de a_i . Par exemple le très célèbre nombre Pi s'écrit avec une infinité de coefficients:

$$\pi = 3 + \frac{1}{7 + \frac{1}{15 + \frac{1}{1 + \dots}}}$$

En pratique on écrira pour simplifier $\pi = [3, 7, 15, 1, 292, \dots]$.

Question 1. (/1) Proposer un type `frac_cont` pour représenter une fraction continue.

Question 2. (/2) Définir en utilisant une fonction anonyme un opérateur de composition `>>` en OCaml qui permet de faire la composition de deux fonctions, c'est-à-dire tel que `g>>f` représente la fonction $g \circ f$.

Question 3. (/1) Préciser le profil de cet opérateur:

Question 4. (/4) À l'aide uniquement de cet opérateur et de fonctions du module `List`, implémenter une fonction `dernier` qui renvoie le dernier élément d'une liste et une fonction `sans_dernier` qui renvoie la liste privée de son dernier élément. Vous définirez comme une application partielle: **la variable qui représente la liste ne doit pas apparaître.**

Question 5. (/4) Implémenter une fonction `fc_vers_R` qui convertit une fraction continue en un Réel. Par exemple:

```
# fc_vers_R [3.;7.;15.;1.;292.];;  
  
- : float = 3.14159265301190249
```

Votre fonction devra utiliser un schéma de type `fold_right` et vous pouvez utiliser `dernier` pour extraire la graine ainsi que `sans_dernier` pour appeler votre schéma.

Question 6. (/3) Implémenter maintenant une fonction `couper n l` qui renvoie les n premiers éléments d'une liste.

Question 7. (/2) Dédurre des fonctions `couper` et `fc_vers_R` une fonction `approx` qui renvoie l'approximation de d'un réel en coupant le développement en fraction continue au nième terme. Vous utiliserez l'opérateur de composition défini ci-dessus et vous définirez comme une application partielle.

Certains réel ont un développement en fraction continue particulier, c'est le cas par exemple de la contante d'euler e .

$e = [2.0; 1.0; 2.0; 1.0; 1.0; 4.0; 1.0; 1.0; 6.0; 1.0; 1.0; 8.0; 1.0; 1.0]$, le premier terme est 2 et ensuite il s'agit d'une répétition de triplet du type $(1, 2n + 1, 1)$. On rappelle que la fonction `List.init` permet de générer une liste d'éléments à partir d'une liste d'entiers, par exemple:

```
# List.init 10 (fun k -> 2*k);;  
  
- : int list = [0; 2; 4; 6; 8; 10; 12; 14; 16; 18]
```

Question 8. (/4) À l'aide de la fonction `List.init` implémenter une fonction `e_fc` qui génère les n premiers termes du DFC de e , attention à bien renvoyer une liste de `float`, dans le bon sens et qui commence par 2!

Rappel sur les fonctions du module **List**

```
List.fold_left;;  
- : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
```

```
List.fold_right;;  
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
```

```
List.hd;;  
- : 'a list -> 'a = <fun>
```

```
List.tl;;  
- : 'a list -> 'a list = <fun>
```

```
List.map;;  
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
```