Séance 9: Evaluation paresseuse

L3 – Université Nice Sophia Antipolis

Exercice 1 (Booléens paresseux, ⋆)
On définit les types suivants :

```
type booleen = Vrai | Faux
type lazybool = booleen Lazy.t
```

1. *Définissez la fonction* `ou : lazybool -> lazybool -> lazybool` *qui calcule le ou logique de deux Booléens paresseux. La fonction renvoie un Booleen paresseux. Il y a plusieurs approches, on en prendra une dans laquelle le deuxième Booléen n'est pas systématiquement forcé.*
2. *Combien de* `!` *sont affichés par le code suivant ?*

```
let vrai = lazy(print_string "!"; Vrai)
let faux = lazy(print_string "!"; Faux)
let _ = Lazy.force (ou (ou faux faux) (ou vrai vrai))
```

□

Dans la suite du TP, on manipule les flots paresseux définis en cours :

```
type 'a item = Item of 'a * 'a flot
and 'a flot = 'a item Lazy.t
let scons x f : 'a flot = lazy(Item(x,f))
let get_item = function (lazy(Item(hd,tl)) : 'a flot) -> (hd,tl)
```

Tous les flots que vous aurez à définir sont des flots infinis d'entiers.

Exercice 2 (scons, ⋆)
Considérons les deux fonctions suivantes

```
let rec ints_from_strict n = scons n (ints_from_strict (n+1))
let rec ints_from n = lazy(Item (n, (ints_from (n+1))))
```

1. *Que se passe-t-il si on réalise l'appel* `ints_from_strict 0` *?*
2. *Que se passe-t-il si on réalise l'appel* `ints_from 0` *?*
3. *Pourquoi le comportement diffère alors qu'on a remplacé l'appel à* `scons` *par sa définition ?*

□

Exercice 3 (Fonctions utilitaires, ⋆⋆)

1. *Définissez les fonctions* `shead` *et* `stail` *qui renvoient la tête et la queue d'un flot.*
2. *Définissez la fonction* `snth` *qui renvoie le n-ième élément d'un flot.*
3. *Définissez la fonction* `sfilter p s` *qui renvoie le flot contenant les éléments de s qui vérifient p.*

4. *Définissez la fonction* `smap2` *telle que* `smap2 f s1 s2` *renvoie le flot contenant les images élément par élément des flots* `s1` *et* `s2` *par la fonction* `f`.
5. *Déduisez-en les opérateurs* `++` *et* `**` *qui additionnent et multiplient deux flots élément par élément.*
6. *Définissez la fonction* `siter n f s` *qui appelle la fonction* `f:int->unit` *sur les n premiers éléments de s.*
7. *Définissez la fonction* `sprint n s` *qui affiche les n premiers entiers du flot s. Par exemple,* `sprint 10 (ints_from 0)` *affiche* `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...]`

□

Exercice 4 (Flots par récurrence, ⋆⋆)
À l'aide d'une fonction récursive, définissez les flots suivants :

1. *le flot des entiers impairs* `[1, 3, 5, 7, ...]`
2. *le flot des factorielles* `[1, 1, 2, 6, 24, 120, 720, ...]`
3. *le flot des entiers de Fibonacci* `[0, 1, 1, 2, 3, 5, 8, 13, 21, ...]`

□

Exercice 5 (Flots par point fixe, ⋆⋆)
À l'aide d'une équation de point fixe, définissez les flots suivants :

1. *le flot constant à 1* `[1, 1, 1, 1, ...]`
2. *le flot des entiers naturels* `[1, 2, 3, 4, ...]`
3. *le flot des factorielles* `[1, 1, 2, 6, 24, 120, ...]`
4. *le flot des entiers de Fibonacci* `[0, 1, 1, 2, 3, 5, 8 , 13, 21, ...]`

□

Exercice 6 (Nombres premiers, ⋆⋆)

1. *Définissez une fonction* `eratosthene s` *qui réalise le crible d'Eratosthene sur le flot* `s` *: la tête du flot est préservée, on retire tous ses multiples dans la queue du flot, et on applique récursivement* `eratosthene`.
2. *Déduisez-en une définition du flot des nombres premiers* `[2, 3, 5, 7, 11, ...]`.

□