HLIN403 Programmation applicative
TD10 : Les avantages du fonctionnel pur – Eléments de Dataflow (map, filter, accumulate) – Les flots (stream)

A. Chateau {annie.chateau@umontpellier.fr}
V. Boudet {vincent.boudet@umontpellier.fr}
H. Chahdi {hatim.chahdi@umontpellier.fr}

1 Eléments de Dataflow (map, filter, accumulate)

Exercice 1 *Que fait l'algorithme A défini ci-dessous ?*

```
Algorithme A
(define (sum-odd-squares tree)
   (cond ((null? tree) 0)
         ((not (pair? tree))
               (if (odd? tree) (square tree) 0))
         (else (+ (sum-odd-squares (car tree))
                  (sum-odd-squares (cdr tree))))))
```

Exercice 2 *Quel est le résultat de l'évaluation de l'expression suivante ?*

```
(sum-odd-squares (list 1 (list 2 (list 3 4) 5) (list 6 7)))
```

Exercice 3 *Que fait l'algorithme B défini ci-dessous ?*

```
Algorithme B
(define (sum-odd-squares2 tree)
        (accumulate +
                    0
                   (map square
                        (filter odd?
                        (enumerate-tree tree)))))
```

Exercice 4 *Quel est le résultat de l'évaluation des expressions suivantes ?*

```
(sum-odd-squares2 (list 1 (list 2 (list 3 4)) 5))

(sum-odd-squares2 (list 1 (list 2 (list 3 4) 5) (list 6 7)))
```

Exercice 5 *Ecrire un algorithme qui calcule le produit des carrés (fonction square) qui sont impairs (prédicat* `odd?`*) des feuilles d'un arbre (nommé* `tree`*) qui ne contient que des entiers positifs.*

2 Stream

Exercice 6 *Compléter les définitions suivantes :*

1. *Un flot de "1" :*

```
(define ones (cons-stream 1 ???))
```

2. *? ? ?*

```
(define (add-streams s1 s2)
        (stream-map + s1 s2))
```

3. *Un flot d'entiers $\geq$1 :*

```
(define integers (cons-stream 1 (add-streams  ?  ??)))
```

4. *Les nombres de Fibonacci*

```
(define fibs (cons-stream 0
                  (cons-stream 1
                        (add-streams (??? fibs) fibs))))
```

Exercice 7 *Définir une fonction* `mul-streams`*, analogue à* `add-streams`*, qui produit le produit deux à deux des éléments de deux flots passés en entrée. Utiliser cette fonction, ainsi que le flot des entiers, pour compléter la défintion suivante du flot dont le nième élément (en partant de 0) est la factorielle de $n+1$:*

```
(define factorials (cons-stream 1 (mul-streams <??> <??>)))
```