4. Collections Fonctionnelles

Jean-Luc Falcone

August 29, 2013

1 Matin

1.1 Min/Max

- Ecrire une fonction `minMax` retournant le plus grand et le plus petit élément d'une liste d'entiers. **Ne pas** utiliser les métodes `min` et `max` de la classe `List`. Cette méthode effectuera un nombre optimal de comparaisons.

1.2 Nombres aléatoire

- Ajoutez la méthode `randomList( n: Int ): (List[Int],RNG)` qui produit une liste de `n` nombres aléatoires.

2 Après-midi

2.1 Utilisation des collections standards

- Etudiez le fichier `Person.scala`
- Remplacer toutes les occurences de `???` par une implémentation respectant les commentaires inclus dans le fichier.

2.2 Arbres binaires

A partir de votre implémentation ou du corrigé, ajoutez les méthodes suivantes aux arbres définis dans la série 3:

- `toList: List[A]`: qui retourne en liste incluant tous les éléments (l'ordre doit correspondre à un parcours en profondeur de gauche à droite.
- `foreach( body: A=>Unit ): Unit` qui applique un effet de bord (`body`) à chaque élément.
- `map[B]( f: A=>B ): Tree[B]` qui produit un nouvel arbre en appliquant à une fonction (`f`) sur chaque élément. La structure doit être préservée.

- `filter( p: A=>Boolean ): Tree[A]` qui supprime tous les noeuds qui ne sont pas évalués à vrai par le prédicat (`p`). L'arbre résultant ne doit pas contenir de noeud pointant sur deux feuilles vides. Dans ce cas ces noeud doivent être remplacé par des feuilles vides.

2.3 Pile mutable

A partir de votre implémentation ou du corrigé. Ajoutez les méthodes suivante aux piles définies dans la série:

- `toList: List[A]` qui retourne une liste scala incluant tous les éléments de la pile. Les éléments au sommet de la pile doivent apparaître au début de la liste. La pile initiale ne doit pas être modifiée.
- `map[B]( f: A=>B ): Stack[B]` qui produit une nouvelle pile en appliquant la fonction (`f`) à chaque élément. L'ordre des éléments correspond à ceux de la pile initiale. Celle-ci ne doit pas être modifiée.

2.4 Foncteur

- Ecrire une fonction qui prend comme paramètres la fonction `f: (A)=>B` et la transforme en fonction `(List[A])=>List[B]`. Cette fonction applique `f` sur chaque élément d'une liste.