INGI1121 – Méthodes de conception de programmes
Devoir 2 — 1, 2, 3… Arbres !

Guillaume Maudoux, Xavier Gillard et Charles Pecheur

Q2 2018–2019

Ce travail est à rendre sur Moodle pour le mardi 30 avril à 18h au plus tard sous la forme d'un zip contenant un fichier `code.dfy` et d'un document `spec.pdf`.

Dans ce devoir, nous allons explorer en détail les arbres 2-3 utilisés pour représenter des séquences d'entiers ordonnés. Un arbre 2-3 est un arbre ordonné, équilibré, terminé par des feuilles vides et pouvant contenir dans chaque nœud une ou deux valeurs et respectivement deux ou trois sous-arbres.

Pour implémenter ces arbres, nous utilisons le type de données suivant :

```
datatype Tree = Vide
              | Deux(Tree, int, Tree)
              | Trois(Tree, int, Tree, int, Tree)
```

On vous demande

- Les invariants de représentation `ok()` ;
- La fonction d'abstraction `abs()` ;
- la spécification des méthodes suivantes
  - `insert(t: Tree, i: int) returns (t': Tree)` qui insère l'élément `i` dans l'arbre `t` et
  - `join(t1: Tree, t2: Tree) returns (t: Tree)` qui fusionne `t1` et `t2` en maintenant l'ordre des éléments.
- l'implémentation de `insert` en Dafny.

La description complète de `ok` nécessite d'introduire plusieurs fonctions et/ou prédicats auxiliaires et l'implémentation de `insert` sur les valeurs de type `Tree` nécessite de trouver comment remonter un nœud au niveau supérieur en cas de rééquilibrage.

En cas de doute, pensez à consulter le livre de référence du cours d'Algorithmes et structures de données [1], et la page Wikipédia anglophone `https://en.wikipedia.org/wiki/2-3_tree`.

S'il est demandé d'écrire `insert` en Dafny, il est en revanche *fortement déconseillé* d'implémenter les fonctions `ok` et `abs` en Dafny. Il est même déconseillé de tenter toute forme de spécifications. Pour ce devoir, elles ne sont pas nécessaires, et vont vous faire perdre beaucoup de temps. Préférez pour ces parties un éditeur de texte adapté, ou une copie manuscrite comme pour le devoir 1.

Pour pouvoir tester votre code, vous pouvez utiliser une fonction Main comme suit

```
method Main() {
    var t := insert(Vide, 1);
    print t, "\n";
}
```

en exécutant ce code (clic droit dans vscode, et "Dafny : Compile and Run (F5)"), cet exemple devrait produire `Tree.Deux(Tree.Vide, 1, Tree.Vide)`.

Enfin, vous aurez certainement besoin de l'instruction `match` en Dafny. Pensez à consulter les ressources disponibles sur Moodle.

1. Sedgewick, R., & Wayne, K. (2007). Algorithms and data structures. Princeton University, COS, 226.