université de BORDEAUX	**Année 2017/2018** — L2 Info/MI, UE 4TIN402U Algorithmique des structures de données arborescentes Vendredi 30 Mars 2018, 8h–9h15 (Durée : 1h15) Documents : **non autorisés** Le barème est donné à titre **indicatif**	**Collège Sciences et Technologies**

Exercice 1 *Algorithmes d'insertion (5 points)*

1. Dessiner l'***arbre binaire de recherche*** construit par ajouts successifs des éléments de la suite

   $$17,\ 25,\ 14,\ 13,\ 19,\ 12,\ 21,\ 22$$

   en utilisant la méthode vue en cours (adjonction aux feuilles).
2. De manière générale, exprimer à l'aide de la notation $O()$ la complexité, dans le pire des cas, de l'algorithme de recherche dans un *arbre binaire de recherche* contenant $n \geqslant 1$ éléments.
3. Dessiner l'***arbre AVL*** construit par ajouts successifs des éléments de la suite de la question 1, en utilisant la méthode vue en cours (adjonction aux feuilles et rééquilibrage). Identifier les éléments de la liste dont l'insertion nécessite un rééquilibrage. Pour ces éléments-là, dessiner l'arbre avant et après le rééquilibrage.
4. De manière générale, exprimer à l'aide de la notation $O()$ la complexité, dans le pire des cas, de l'algorithme de recherche dans un *arbre AVL* contenant $n \geqslant 1$ éléments.

Exercice 2 *Affichage d'arbre (4 points)*

Dans cet exercice, on travaille sur des arbres binaires donnés par le type :

```
type 'a tree = Empty | Bin of ('a * 'a tree * 'a tree)
```

On se propose d'écrire une fonction `string_of_int_tree : int tree -> string` qui convertit un arbre en une chaîne de caractères qui le représente en syntaxe OCaml. Par exemple :

— `string_of_int_tree Empty` doit retourner la chaîne `"Empty"`,

— `string_of_int_tree (Bin(42,Bin(2,Empty,Empty),Empty))` doit retourner la chaîne `"Bin(42,Bin(2,Empty,Empty),Empty)"`.

La fonction `string_of_int : int -> string` de la bibliothèque standard convertit un entier de type `int` en une chaîne de caractères qui le représente. Ainsi, `string_of_int (-34)` renvoie la chaîne `"-34"`. Enfin, la concaténation de chaînes est notée `^`. Par exemple, `"abc" ^ "def"` renvoie `"abcdef"`.

1. Écrivez la fonction `string_of_int_tree`.
2. Utilisant la notation $O()$, évaluer le nombre de fois où l'opérateur `^` est utilisé, dans le cas le pire, lors de l'appel de la fonction `string_of_int_tree` sur un arbre à n nœuds.

Exercice 3 *Recherche du successeur dans un ABR (4 points)*

Écrire une fonction `succ` de type `('a * int) tree -> int -> int` telle que si `t` est un arbre binaire de recherche et `key` est une clé, `succ t key` renvoie la plus petite clé de `t` strictement supérieure à `key`. Par exemple, soit l'arbre binaire de recherche `t` défini ainsi :

```
let t = Bin((45,1),
                  Bin((17,1),
                       Bin((15,1), Empty, Empty),
                       Bin((35,2), Empty, Empty)),
                  Bin((84,1),
                       Empty,
                       Bin((99,3), Empty, Empty)))
```

La suite croissante des clés contenues dans cet arbre est $15, 17, 35, 45, 84, 99$. Pour la clé `key=45`, la valeur renvoyée doit être `84`.

Remarque. Pour une clé supérieure ou égale à la plus grande clé de `t`, `succ` n'est pas défini. La fonction `succ` doit traiter ces cas d'erreur (on pourrait utiliser `failwith`).

Exercice 4 *Collage en branche gauche (4 points)*

1. Écrire une fonction OCaml qui, à partir de deux arbres binaires **pleins et non vides** `t1` et `t2`, retourne l'arbre `t` obtenu en ***remplaçant la feuille*** de `t1` qui se trouve au bout de sa branche la plus à gauche, par `t2`.
2. Donner une condition suffisante sur les arbres `t1` et `t2` pour que l'arbre résultat `t` soit un arbre binaire de recherche.

Exercice 5 *Preuve de propriété (3 points)*

Soit t un arbre binaire. On note $i(t)$ le nombre de nœuds internes de t et $f(t)$ son nombre de feuilles. Montrer la propriété suivante (vous pouvez utiliser une récurrence) :

Si $f(t) = i(t) + 1$, alors l'arbre t est plein.