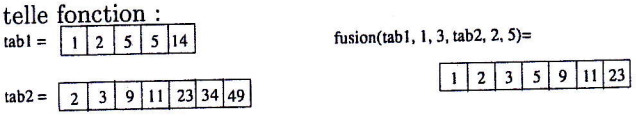
**Université Denis Diderot** **Examen de TO2 – L1**
**2011–2012**

IMPORTANT. Durée **3 heures**. Il sera tenu compte de la **clarté** et de la **qualité** de votre copie. Aucun document autorisé.

## A/ Cours : tri-fusion.

A1- On suppose disposer de deux tableaux d'entiers tab1 et tab2. De plus, on suppose que tab1 et tab2 sont **triés** dans un ordre croissant. Ecrire une fonction `fusion`(tab1, début1, fin1, tab2, début2, fin2) qui renvoie un tableau **trié** de la fusion des contenus des deux tableaux tab1 et tab2 (respectivement de début1 à fin1 et de début2 à fin2) passés en paramètre. L'exemple ci-dessous schématise une telle fonction :




A2- Expliquer comment utiliser alors la fonction `fusion` pour **trier** un tableau quelconque. Donner les codes Java correspondant à vos explications.

A3- Donner la complexité d'un tel tri.

## B/ Arbres binaires.

On considère des arbres binaires (classe Arbre) définis à l'aide des classes suivantes :

```
public class Noeud {
    private char etiquette;
    private Noeud gauche;
    private Noeud droit;
        public Noeud (char etiquette, Noeud gauche, Noeud droit) {
            this.etiquette = etiquette;
            this.gauche = gauche;
            this.droit = droit;
        }
}
class Arbre {
    private Noeud racine;
    public Arbre () { racine = null; }
}
```

B1- Écrivez, dans la classe `Arbre`, une méthode `void aDevientB()` qui remplace chaque caractère `'a'` des étiquettes par un `'b'`.

B2- Écrivez, dans la classe `Arbre`, une méthode `void afficheFeuilles()` qui affiche les feuilles de gauche à droite, par exemple sur l'arbre de la Figure 1, la méthode doit afficher "bonjour".

B3- Écrivez, dans la classe `Arbre`, une méthode `void enleveFeuilles()` qui élimine toutes les feuilles de l'arbre, par exemple sur l'arbre de la Figure 1, doit retourner l'arbre de la Figure 2, si l'arbre était vide, il restera vide. Si l'arbre n'a qu'un nœud (qui est donc une feuille) il deviendra vide.

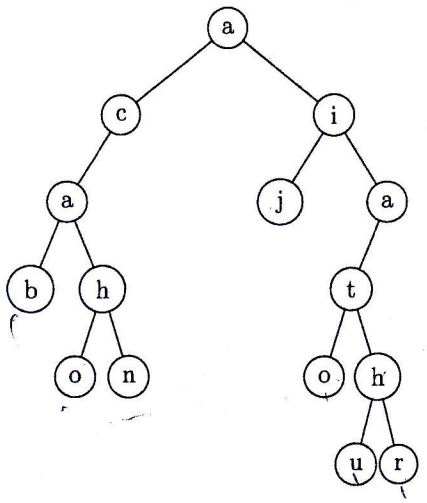



FIGURE 1 – arbre 1

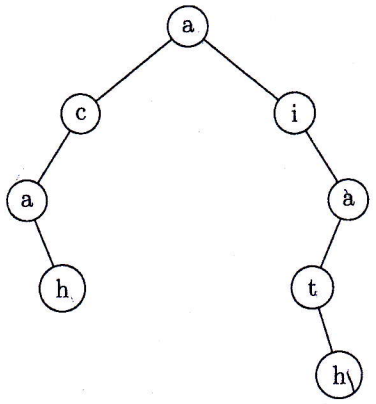



FIGURE 2 – arbre 2

**C/ Arbres binaires récursion et complexité.**

C1- Dessiner les **neuf** arbres binaires avec 0, 1, 2, et 3 nœuds.

C2- Expliquer comment obtenir *tous* les arbres binaires avec $n$ ($n \geq 1$) nœuds à partir des arbres binaires avec $k$ nœuds et $n-1-k$ nœuds. En particulier, donner toutes les valeurs possibles de $k$.

C3- En déduire un algorithme (en français) récursif `touslesArbres` construisant tous les arbres binaires avec $n$ nœuds. Expliciter clairement les *conditions d'arrêt* et le *corps de la récursion* dans votre algorithme.

C4- Soit $T_n$ la complexité de la fonction `touslesArbres` de la question précédente. On suppose que $T_0 = 0$ et $T_1 = 1$ (i.e., la complexité de la construction d'un arbre vide est 0 et celle d'un arbre avec un nœud est 1). Montrer que

$$T_n = 1 + \sum_{k=0}^{n-1} (T_k + T_{n-1-k}) \ .$$

(Suggestion : on pourra s'appuyer sur des schémas commentés.)

**D/ Listes et matrices creuses.** Dans cet exercice, on considère des matrices d'entiers. Une matrice **creuse** est une matrice essentiellement composée de 0. Seules quelques valeurs sont donc non nulles. A titre d'exemple, voici une matrice creuse

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 0 | 0 | 0 |
| 1 | 0 | -3 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 6 | 0 |

Notons que l'élément situé ligne 4, colonne 4 vaut 6. On souhaite représenter une matrice creuse avec une liste chaînée constituée des éléments non-nuls avec leurs indices, donc la matrice ci-dessus pourrait être représentée par :

$$[(2,(1,2)); (1,(2,1)); (-3,(2,3)); (6,(4,4)); (0,(4,5))]$$

Les éléments sont rangés par indice croissant (par ligne puis par colonne). Le dernier élément de la liste (même nul) permet de connaître les indices maximaux de la matrice (ici l'élément "(0, (4,5))" permet de savoir qu'il y a 4 lignes et 5 colonnes dans la matrice).