# IR1 Algorithmique Groupe 2, 2002. Durée 30 minutes.

**Nom :**
**Prénom :**
On considère les types suivants permettant de manipuler des arbres binaires dont les étiquettes des nœuds sont des caractères.

```
typedef struct noeud {
  char e;
  struct noeud *fg, *fd;
} Noeud;
typedef Noeud * Arbre;
```

On dit qu'un arbre est *bien équilibré* si *pour tout nœud*, la différence entre la hauteur de son sous-arbre gauche et la hauteur de son sous-arbre droit est comprise entre -1 et 1.

**Exercice 1.**— Écrire une fonction `hauteur(Arbre a)` qui calcule la hauteur de l'arbre `a`. La hauteur de l'arbre vide sera -1.

**Exercice 2.**— Écrire une fonction `int equilibre(Arbre a)` qui renvoie 1 si l'arbre `a` est bien équilibré, et 0 sinon.

On considère un nouveau type d'arbres (`Arbre2`) tel que chaque nœud contient une étiquette de type caractère et un autre champ `info` de type `int` destiné à stocker une information.

```
typedef struct noeud2 {
  char e;
  int info;
  struct noeud2 *fg, *fd;
} Noeud2;
typedef Noeud2 * Arbre2;
```

**Exercice 3.**— Écrire une fonction `void calcul(Arbre2 a)` qui prend en argument un arbre de type `Arbre2` dont les champs `info` sont tous égaux à zéro, et renvoie cet arbre avec le champ `info` de chaque nœud égal à la hauteur du sous-arbre de racine ce nœud.

**Exercice 4.**— Écrire une nouvelle fonction `int equilibre2(Arbre2 a)` qui renvoie 1 si l'arbre `a` est bien équilibré, et 0 sinon.

**Exercice 5.**— Quelles sont les complexités en temps des fonctions `equilibre` et `equilibre2` ?