| | | |
|---|---|---|
| efrei PARIS SUD Ingénieurs en mouvement<br>2013-2014 | L'3 – UE « Structures de données » (SDD)<br>Evaluation de TD – **Groupe B**<br>Mercredi 11 décembre 2013<br>Durée 30 mn – Sans document ni équipement électronique | A. Morelle |

# ÉNONCÉ – FEUILLE DE RÉPONSE

**NOM** …… **Prénom** ……

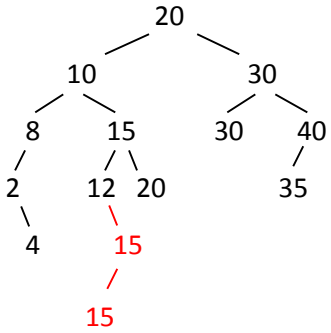
*Les types de données considérés sont ainsi définis :*

```
type structure nœud
    info : entier
    sag : adresse nœud          // sous-arbre gauche
    sad : adresse nœud          // sous-arbre droit
fintype

type adresse nœud : arbre
```

*Les algorithmes demandés sont typiquement attendus en langage algorithmique et devront être autonomes.*

**Exercice 1.**




| | |
|---|---|
| Cet arbre est-il un *arbre binaire de recherche* (au sens de la définition donnée en TD) ?<br>Justifier sommairement. | OUI - NON (entourer la bonne réponse)<br><br>En chaque nœud :<br>. toutes les valeurs du sous-arbre gauche ≤ valeur du nœud<br>. toutes les valeurs du sous-arbre droit > valeur du noeud |
| On fait appel deux fois à l'algorithme classique d'insertion pour insérer deux fois la valeur 15 dans cet arbre. Indiquer précisément dans l'arbre ci-dessus où seront créés ces deux nouveaux nœuds de valeur 15. | |
| On souhaite afficher les valeurs de tous les nœuds de l'arbre. Qu'afficherait *un parcours en profondeur préfixe (ou pré-ordre)* ? | 20 10 8 2 4 **15** **12** **20** 30 30 40 35 |

**Exercice 2.**

Procédure afficherPDG( a : arbre, p : EntierNat )
// Donnée : un arbre quelconque a dont il s'agit d'afficher, dans l'ordre droite-gauche, les valeurs des nœuds de
//  profondeur p.
//  *Exemple. Avec l'arbre de l'exercice 1 et p=3, cette procédure doit afficher : 35 20 12 2*
// Donnée : la profondeur p considérée, de type entier naturel
// Variables locales : …. (si besoin)
Début
  Si a = NULL alors retourner ;
  // Sinon
  si p = 0 alors
    afficher(a-→info) ;
    retourner ;
  finsi
  afficherPDG(a→sad, p-1) ;
  afficherPDG(a→sag, p-1) ;
Fin.

**Exercice 3.** On adopte la définition suivante : « *un arbre binaire est parfaitement équilibré ssi, en tout nœud, la différence entre le nombre de nœuds du sous-arbre gauche et le nombre de noeuds du sous-arbre droit est au plus de 1* ».

Fonction estEquilibré( a : arbre, n : EntierNat ) : booléen
// Donnée : un arbre quelconque a
// Donnée modifiée : un entier naturel n dont la valeur finale est nombre de nœuds de l'arbre a s'il est parfaitement
//  équilibré, ou indéfinie sinon.
// Résultat : vrai ssi l'arbre a est parfaitement équilibré au sens de la définition ci-dessus.
// Variables locales : …. (si besoin)
Début
  Si a = NULL alors
    n ← 0 ; retourner vrai ;
  finsi
  // Sinon
  Si estEquilibré(a→sag, ng) = faux alors retourner faux
  Si estEquilibré(a→sad, nd) = faux alors retourner faux
  Si | ng – nd | > 1 alors retourner faux
  n ← ng + nd + 1 ;
  retourner vrai
Fin.