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

ÉNONCÉ – FEUILLE DE RÉPONSE

NOM …… **Prénom** ……

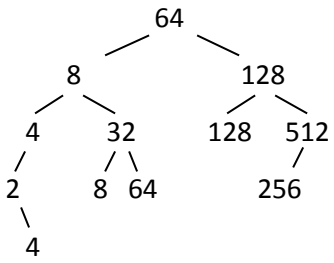
Les types de données considérés sont ainsi définis :

```
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) ?	OUI - (NON) (entourer la bonne réponse)
Si non, remplacer dans l'arbre chaque valeur qui pose problème par la plus petite valeur qui peut convenir, ou indiquer ci-contre toute autre raison du problème.	8 fils gauche de 32 : à remplacer par 9
On souhaite afficher les valeurs de tous les nœuds de l'arbre. Qu'afficherait *un parcours en profondeur post-fixe (ou post-ordre)* ?	4 2 4 16 64 32 8 128 256 512 128 64

Exercice 2.

Procédure dupliquer(a : arbre, b : arbre)
// Donnée : un arbre quelconque a à dupliquer
// Donnée modifiée : l'arbre b duplication de l'arbre a
// Variables locales : …. (si besoin)
Début
    Si a = NULL alors alors
        b ← NULL ; retourner ;
    finsi
Fin.

// Sinon

b ← réserver nœud

b→info ← a→info

dupliquer(a→sag, b-→sag)

dupliquer(a→sad, b→sad)

Exercice 3.

Fonction profminx(a : arbre, x : entier, p : EntierNat) : entier
// Donnée : un arbre quelconque a
// Donnée : un entier x
// Donnée : un entier naturel p correspondant à la profondeur courante (0 au premier appel)
// Résultat : la profondeur du nœud de valeur x le moins profond, ou -1 si aucun.
// *Exemple. Si on prend l'arbre de l'exercice 1 et x=20, la profondeur du nœud de valeur x le moins profond*
// *est 1.*
// Variables locales : …. (si besoin)
Début
    Si a = NULL alors retourner -1
Fin.

// Sinon

Si a→info = x alors retourner p

pg ← profminx(a→sag, x, p+1)

pd ← profminx(a→sad, x, p+1)

si pg ≠ -1 et pd ≠ -1 alors retourner min(pg, pd) + 1
si pg ≠ -1 alors retourner pg + 1
retourner pd + 1