Algorithmique
Correction Partiel n° 1

Info-Sup – Epita

24 jan. 2012

Des arbres binaires

Solution 1 **(Occurrences –** ***2 points*****)**

1. L'arbre est celui de la figure 1

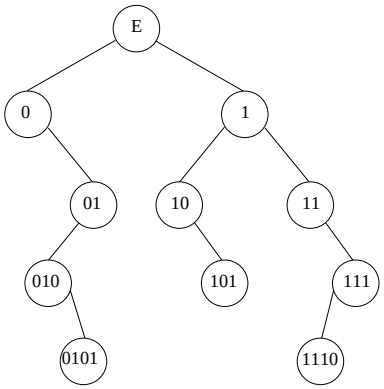

Figure 1 – Arbre B.

2. LCI(B)=14, PME(B)=3.66

Solution 2 **(Incontestablement... –** ***3 points*****)**

1. L'arbre est celui de la figure 2

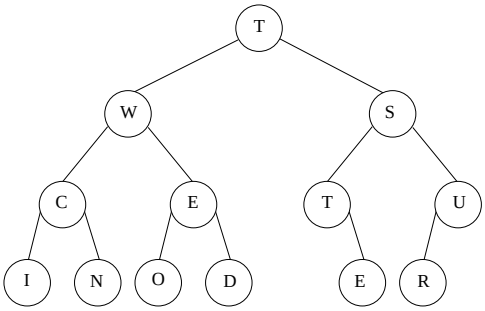

Figure 2 – Arbre "IN CODE WE TRUST".

2. Le parcours *suffixe* de l'arbre `B` donne : `I N C O D E W E T R U S T`, ce que je vous laisse traduire.

Solution 3 **(AB Somme - *4 points*)**

1. **Principe :** On utilise un algorithme basé sur le parcours profondeur main gauche et la définition récurrente de la taille d'un arbre binaire. C'est à dire que la somme calculée pour un arbre vide sera 0 et pour un arbre non vide sera la valeur de l'étiquette du noeud augmentée du poids du sous-arbre gauche (*appel récursif*) et du poids du sous-arbre droit (*appel récursif*).

2. **Algorithme :**

```
algorithme fonction calcule_poids : entier
parametres locaux
   ArbreBinaire B

debut
   si B = arbre-vide alors
      retourne 0
   sinon
      retourne contenu(racine(B)) + calcule_poids(g(B)) + calcule_poids(d(B))
   fin si
fin algorithme fonction calcule_poids
```

Et des listes

Solution 4 **(Liste contiguë : suppression – *6 points*)**

Spécifications :
La fonction `supprime (x, L)` supprime la première occurrence de la valeur x dans la liste triée L. Elle retourne un booléen indiquant si la suppression a eu lieu.

```
algorithme fonction supprime : booleen
    parametres locaux
        t_element    x
    parametres globaux
        t_liste      L

    variables
        entier    i, j
debut
    i ← 1                                              /* recherche de x */
    tant que (i <= L.longueur) et (x > L.elts[i]) faire
        i ← i+1
    fin tant que
    si (i > L.longueur) ou (x < L.elts[i]) alors
        retourne faux
    sinon                                              /* décalages */
        pour j ← i+1 jusqu'a L.longueur faire
            L.elts[j-1] ← L.elts[j]
        fin pour
        L.longueur ← L.longueur - 1
        retourne vrai
    fin si
fin algorithme fonction supprime
```

Solution 5 **(Liste : progression arithmétique – *5 points*)**

Spécification :

La fonction `arithmetique (t_pListe L) : entier` vérifie si la liste L suit une progression arithmétique. Elle retourne la valeur de la raison si c'est le cas, la valeur 0 sinon (de même si la liste contient moins de 2 éléments).

Principe :

Si la liste a au moins deux éléments, la raison potentielle est donnée par la différence entre les deux premiers éléments. On parcourt la liste tant que l'élément suivant est égal à l'élément courant ajouté à la raison. Si le parcours atteint le dernier élément de la liste alors la propriété est vraie, fausse sinon. A noter que si les deux premiers éléments sont égaux, le parcours n'est pas effectué.

```
algorithme fonction arithmetique : entier
    parametres locaux
        t_liste    L

    variables
        entier     raison, i

debut
    raison ← 0
    si (L.longueur > 1) ou (L.elts[2] - L.elts[1] <> 0) alors
        raison ← L.elts[2] - L.elts[1]
        i ← 2
        tant que (i < L.longueur) et (L.elts[i] + raison = L.elts[i+1]) faire
            i ← i+1
        fin tant que
        si i < L.longueur alors
            raison ← 0
        fin si
    fin si
    retourne raison
fin algorithme fonction arithmetique
```