Algorithmique
Partiel nº 1

Info-Sup – Epita

D.S. 311897.31 BW (24 jan 2012 - 10 :00)

Remarques (à lire !) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Les algorithmes :**
  - Tout algorithme doit être écrit dans le langage Algo (pas de C#, Caml ou autre).
  - Tout code Algo non indenté ne sera pas corrigé.
  - En dehors d'indication dans les énoncés, vous ne pouvez utiliser aucune routine (fonction ou procédure) supplémentaire.
  - Tout ce dont vous avez besoin (opérations de types abtraits, types) est donné en annexe.
  - **Rappel :** pour chaque algorithme, lorsque demandé sur les feuilles de réponses, vous devez donner :
    - **Les spécifications** c'est à dire ce qu'il fait, les paramètres (types et significations) et les éventuelles conditions d'utilisation.
    - **Le principe algorithmique** c'est à dire en clair la méthode retenue pour résoudre le problème.

    Attention le principe et les spécifications sont notés, ne pas les envisager revient à sacrifier directement des points.
- ☐ Durée : 2h.

Des arbres binaires

Exercice 1 (Occurrences – *2 points*)

Soit l'arbre B défini, sous forme hiérarchique de la manière suivante :

```
B = {E,0,1,01,10,11,010,101,111,0101,1110}
```

1. Représenter l'arbre B graphiquement.
2. Déterminer la longueur de cheminement interne et la profondeur moyenne externe de l'arbre B.

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

Soit l'arbre B dont les parcours *préfixe* et *infixe* affichent les séquences suivantes :

```
Préfixe : T W C I N E O D S T E U R
Infixe  : I C N W O E D T T E S R U
```

1. Représenter graphiquement l'arbre B correspondant à ces deux parcours.
2. Donner le parcours *suffixe* de l'arbre B.

Exercice 3 (AB Somme - *4 points*)

Définition : Soit un arbre binaire B quelconque étiqueté par des valeurs entières, on appelle ***poids de B*** *la somme de toutes les étiquettes de l'arbre B.*

1. Donner le principe d'un algorithme déterminant le poids de B.
2. En utilisant les opérations définies par le *type algébrique abstrait* rappelé en annexe (dernière page), écrire la **fonction récursive** ***"calcule_poids(B)"*** correspondant à ce principe où B est de type ArbreBinaire.

   Notes :
   - *Vous pouvez déclarer toutes les variables locales supplémentaires que vous jugerez nécessaires.*
   - *Si vous désirez utiliser des opérations supplémentaires, vous devez au préalable les définir abstraitement.*

Et des listes

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

Écrire un algorithme qui supprime la première occurrence de la valeur x dans une liste L (de type `t_liste`) triée en ordre croissant. L'algorithme devra retourner un booléen indiquant si la suppression a pu être effectuée.

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

Écrire une fonction, après avoir donné son principe, qui vérifie si les valeurs entières d'une liste (de type `t_Liste`), contenant au moins deux éléments, suivent une progression arithmétique.

Rappel : Une progression arithmétique est une suite de nombres rangés dans un ordre tel que chacun d'eux s'obtient en ajoutant un nombre constant (la *raison*) à celui qui le précède.

Si la liste a au moins deux éléments et suit bien une progression arithmétique de raison non nulle, la fonction devra retourner la *raison* de la suite. Dans le cas contraire, la fonction devra retourner 0.

Annexes

Le type utilisé pour représenter les listes

Dans les exercices 4 et 5 les listes sont des listes d'entiers représentées en contiguë par le type `t_liste` défini ci-dessous.

```
constantes
    LMax = ...

types
    t_vectLMaxElts = LMax entier

    t_liste = enregistrement
        t_vectLMaxElts   elts
        entier           longueur
    fin enregistrement t_liste
```

Type algébrique abstrait de l'arbre binaire

```
TYPES
    ArbreBinaire

UTILISE
    Noeud, Elément

OPERATIONS
    arbre-vide : → ArbreBinaire
    <_, _, _>  : Noeud × ArbreBinaire × ArbreBinaire → ArbreBinaire
    racine     : ArbreBinaire → Noeud
    g          : ArbreBinaire → ArbreBinaire
    d          : ArbreBinaire → ArbreBinaire
    contenu    : Noeud → Element

PRECONDITIONS
    racine(B) est-défini-ssi B ≠ arbre_vide
    g(B) est-défini-ssi B ≠ arbre_vide
    d(B) est-défini-ssi B ≠ arbre_vide

AXIOMES
    Racine(<o, B1, B2>) = o
    g(<o, B1, B2>) = B1
    d(<o, B1, B2>) = B2

AVEC
    Noeud        o
    ArbreBinaire B, B1, B2
```