Algorithmique
Partiel nº 1

Info-Sup – Epita

D.S. 312053.19 BW (20 jan 2011 - 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.
  - **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.

Exercice 1 (Un peu de cours... – *4 points*)

1. Dans la résolution d'un problème, quelle raison principale motive l'utilisation d'une pile ?
2. Citer 3 sortes d'arbres binaires particuliers. Donner une représentation graphique de chacun d'eux pour un nombre maximum de 13 noeuds (*le minimum étant de 5 noeuds*).
3. Donner le principe d'un algorithme qui utiliserait ***obligatoirement*** une pile pour tester qu'un mot donné sous la forme d'une chaîne de caractères est ou non un palindrome.

Exercice 2 (Cheminons - *6 points*)

Soit le type algébrique abstrait suivant :

```
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
```

1. Donner le principe d'un algorithme déterminant pour un arbre binaire quelconque la longueur de cheminement externe de celui-ci.

2. En utilisant les opérations définies par le *type algébrique abstrait*, écrire la **procédure récursive *"calcule_ lce(B,h,lce)"*** correspondant à ce principe ou B est de type ArbreBinaire, h et lce de type Entier représentant respectivement la hauteur du noeud racine de B et la longueur de cheminement externe de B.

   Notes :
   - *Vous pouvez déclarer tous les variables locales supplémentaires que vous jugerez nécessaires, en gardant à l'esprit que le résultat doit être retourné (*sera contenu*) par le paramètre global* **"lce"**.
   - *Vous donnerez l'appel de cette procédure.*
   - *Si vous désirez utiliser des opérations supplémentaires, vous devez au préalable les définir abstraitement.*

Le type utilisé pour représenter les listes

Dans les exercices qui suivent, les listes sont représentées en contiguë (par le type `t_liste`).

```
constantes
    LMax = ...

types
     /* déclaration du type t_ element*/
    t_vectLMaxElts = LMax t_element

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

Exercice 3 (Minimum - *5 points*)

Après avoir donné son principe, écrire l'algorithme `minimum` qui détermine la position de la valeur minimum dans une liste L entre les positions données d et f (avec $1 \leq d < f \leq L.longueur$).

Exemple :

Dans la liste suivante :

	1	2	3	4	5	6	7	8	9	10	...
L.elts	4	-5	3	-3	8	-2	0	3	-6	7	...

Entre les positions $d = 3$ et $f = 8$, le minimum est à la position 4.

Exercice 4 (Tri par sélection – *5 points*)

1. Écrire la procédure `swap` qui échange le contenu de deux variables de type `t_element`.
2. Écrire un algorithme qui trie en ordre croissant une liste. Votre algorithme **doit obligatoirement** utiliser l'algorithme `minimum` de l'exercice précédent, la procédure `swap` de la question 1.

   Exemple :
   La liste donnée à l'exercice précédent sera après le tri :

	1	2	3	4	5	6	7	8	9	10	...
L.elts	-6	-5	-3	-2	0	3	3	4	7	8	...