Algorithmique
Correction Contrôle n° 2

INFO-SUP – EPITA

7 mar. 2012

Solution 1 **(ABR : chemin de recherche –** ***2 points*****)**

Les séquences ② et ④ sont impossibles :

① 50, on descend à gauche - 15, on descend à droite - 48 on descend à gauche - 22, on descend à droite - 46, on descend à gauche - **42**

② 48, on descend à gauche - 15, on descend à droite - **45, on descend à gauche** - 22, on descend à droite - **47 ne peut se trouver là, il n'est pas inférieur à 45 !**

③ 15, on descend à droite - 22, on descend à droite - 45, on descend à gauche - 35, on descend à droite - **42**

④ **22, on descend à droite** - 45, on descend à gauche - 43, on descend à gauche - **15 ne peut pas se trouver là, il n'est pas supérieur à 22**

Solution 2 **(Trichotomie -** ***7 points*****)**

1. Le principe de la recherche trichotomique est le suivant :

   La recherche de l'élément `x` se fait sur une liste `l` dont les `n` éléments sont compris entre deux bornes : une gauche `g` et une droite `d` initialisées respectivement aux valeurs `1` et `n`.

   Le principe de recherche est récurrent :
   Tant que `g+1<d` (au moins deux valeurs d'écart) on calcule deux valeurs de pivots `p1=(2g+d) div 3` et `p2=(g+2d) div 3`. On regarde alors si `x=ieme(l, p1)` ou si `x=ieme(l, p2)`.
   Si c'est le cas, la recherche est positive et l'on retourne `p1` ou `p2` selon le cas.

   Sinon, on poursuit la recherche de `x` sur un intervalle réduit au tiers des éléments.

   Dans ce cas, la question est : Quel est cet intervalle ?

   Les réponses possibles sont les suivantes :
   (a) si `x<ieme(l, P1)` alors on recommence sur l'intervalle `[g, p1-1]`
   (b) si `ieme(l, P1)<x<ieme(l, P2)` alors on recommence sur l'intervalle `[p1+1, p2-1]`
   (c) si `ieme(l, P2)<x` alors on recommence sur l'intervalle `[P2+1,d]`

   Lorsque les bornes `g` et `d` se croisent, la recherche est négative (l'élément `x` n'existe pas dans la liste). Dans ce cas, on retourne `0`.

   remarque : Nous pourrions continuer la récursion lorsque les bornes ont moins de deux valeurs d'écart, mais nous ferions des tours de plus et des calculs de pivots inutiles. L'idéal est, lorsque

les bornes sont égales ou n'ont qu'une valeur d'écart, de tester la valeur de l'élément directement sur ces bornes. C'est cette solution qui est retenue pour l'algorithme et l'arbre d'exécution demandés.

2. Algorithme :

Cette solution ne tient pas compte de l'aspect débranchant des `Retourne` et positionne à chaque fois les `sinon`. C'est bien évidemment optimisable.

```
algorithme fonction trichotomie : entier
parametres locaux
    element x
    liste   l
    entier  g, d
variables
    entier p1, p2
debut
    si g+1<d alors
       p1 ← (2*g+d) div 3
       p2 ← (g+2*d) div 3
       si x=ieme(l,p1) alors
          retourne p1               /* Recherche positive sur p1 */
       sinon
          si x=ieme(l,p2) alors
             retourne p2            /* Recherche positive sur p2 */
          sinon
             si x<ieme(l,p1) alors
                retourne trichotomie(x, l, g, p1-1)
             sinon
                si x<ieme(l,p2) alors
                   retourne trichotomie(x, l, p1+1, p2-1)
                sinon
                   retourne trichotomie(x, l, p2+1, d)
                fin si
             fin si
          fin si
       fin si
    sinon
       si x=ieme(l,g) alors
          retourne g            /* Recherche positive sur g ou d */
       sinon
          si x=ieme(l,d) alors
             retourne d         /* Recherche positive sur d */
          sinon
             retourne 0         /* Recherche négative */
          fin si
       fin si
    fin si
fin algorithme fonction trichotomie
```

Solution 3 **(Listes chaînées : occurrences – *5 points*)**

Spécifications :

La fonction `occurrences (t_element` x`, t_pListe` L`)` retourne un entier indiquant le nombre d'occurrences de la valeur x dans la liste L.

```
algorithme fonction occurrences : entier
   parametres locaux
      t_element   x
      t_pListe    L

   variables
      entier   nb_x
debut
   nb_x ← 0
   tant que L <> NUL faire
      si L↑.valeur = x alors
         cpt ← cpt +1
      fin si
      L ← L↑.suivant
   fin tant que
   retourne cpt
fin algorithme fonction occurrences
```

Solution 4 **(Listes chaînées : insertion – *6 points*)**

Spécifications :

La procédure `insere (t_element` x`, t_pListe` L`)` insère la valeur x à sa place dans la liste L triée en ordre croissant.

```
algorithme procedure insere
    parametres locaux
        t_element    x
    parametres globaux
        t_pListe     L

    variables
        t_pListe     p, prec, new
debut
                  /* recherche de la place avec conservation du précédent */
    p ← L
    tant que (p <> NUL) et (x > p↑.valeur) faire
        prec ← p
        p ← p↑.suivant
    fin tant que
                  /* ajout  */
    allouer (new)
    new↑.valeur ← x
    new↑.suivant ← p
    si p = L alors          /* insertion en tête */
        L ← new
    sinon
        prec↑.suivant ← new
    fin si
fin algorithme procedure insere
```