CORRECTIONS DES EXERCICES DU CONCOURS D'ENTREE EPITA 2014

Pour toutes les fonctions demandées, plutôt que de fournir des solutions en C, Caml ou Pascal, nous avons préféré fournir une solution algorithmique en pseudo-langage facilement implémentable dans un de ces langages.

A) Chemin de recherche ...

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**

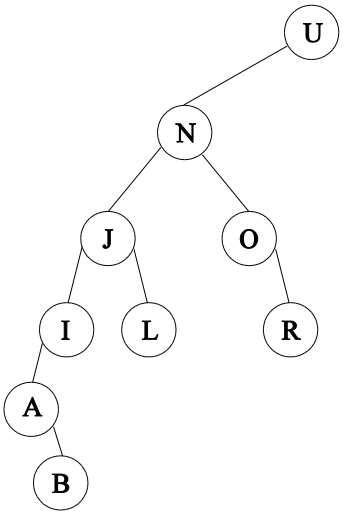
2) **48**, on descend à gauche - 15, on descend à droite - **45, on descend à gauche** - 22, on descend à droite - **47 ne peut se trouver là, il n'est pas inférieur à 45 !**

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

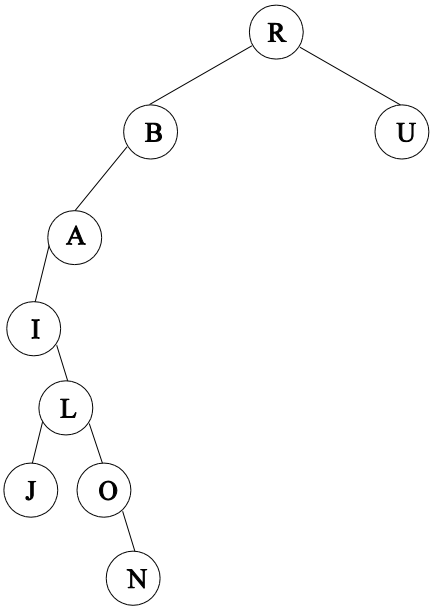
4) **22, on descend à droite** - 45, on descend à gauche - 43, on descend à gauche - **15 n'est pas supérieur à 22**

B) ABR : Insertions...

1) BR obtenu après insertions en feuille :



2) BR obtenu après insertions en racine :



C) ABR : Recherche du minimum …

1) La valeur minimum d'un ABR se trouve au bout de son bord gauche.

2) ***Principe :***
Il suffit de parcourir le bord gauche de l'arbre binaire : tant que le fils gauche n'est pas vide, on descend à gauche. Lorsqu'on arrive à un nœud sans fils gauche, celui-ci contient la valeur minimale de l'ABR.

3) algorithme :

Version itérative :

```
algorithme fonction cherche_min : Element
parametres locaux
   ArbreBinaire B    /* non vide */
debut
   tant que g(B) <> arbre-vide faire
      B <- g(B)
   fin tant que
   retourne (contenu (racine (B)))
fin algorithme fonction cherche_min

 \end{alltt}
```

Version récursive :

```
algorithme fonction cherche_min : Element
parametres locaux
   ArbreBinaire B    /* non vide */
debut
   si g(B) = arbre-vide alors
      retourne (contenu (racine (B)))
   sinon
      retourne (cherche_min (g(B)))
   fin si
fin algorithme fonction cherche_min
```

D) Liste : Suppression post-dichotomie …

Spécifications :
La fonction ***dichopremsupprime (t_element x, t_vectLmaxElts l, entier g, d) : booléen*** recherche l'élément x dans la liste triée l entre les bornes g et d. Elle retourne ensuite un booléen indiquant si la suppression de la première occurrence de l'élément x a bien eu lieu.

Principe :
Dans un premier temps, la fonction recherche la première occurrence dichotomiquement ; c'est-à-dire que l'on calcule la position de l'élément médian, et l'on teste si l'élément x est inférieur ou égal à celui-ci. Si c'est le cas on recommence sur l'intervalle de gauche (en conservant le médian). Si ce n'est pas le cas, on recommence sur l'intervalle de droite (sans conserver l'élément médian).

Une fois l'élément trouvé (s'il existe), nous sommes sûrs d'être sur la première occurrence de celui-ci. Il ne reste plus qu'à faire la suppression de celui-ci en décalant tous ceux qui le suivent d'une case vers la gauche et à retourner Vrai.

S'il n'existe pas, on se contentera de retourner Faux.

```
algorithme fonction dichopremsupprime : booléen
parametres locaux
   t_element e
   t_vectLmaxElts l
   entier g, d
variables
   entier m
début
   si g < d alors
      m ← (g + d ) div 2
      si x <= l.elts[m] alors
         retourne (dichopremsupprime(e, l, g, m))
      sinon
         retourne (dichopremsupprime(e, l, m+1, d))
      fin si
   sinon /* dernier élément de la liste */
   si x <> l.elts[m] alors
      retourne (faux)
   sinon
      tant que (g <= l.longueur-1) faire
         l.elts[g] ← l.elts[g+1]
         g ← g + 1
      fin tant que
      retourne (vrai)
   fin si
fin algorithme fonction dichopremsupprime
```