TD numéro 7
Arbres binaires

• Écrire une fonction booléenne qui teste qu'une liste représente bien un arbre binaire.

• Écrire une fonction booléenne qui teste qu'un arbre est une feuille, c'est-à-dire un nœud qui n'a aucun fils.

• Écrire une fonction qui calcule la hauteur d'un arbre, définie ainsi : la hauteur d'un arbre est 1 + le maximum des hauteurs des sous-arbres (gauche, droit), la hauteur d'une feuille étant zéro.

• Écrire une fonction booléenne qui teste si une valeur appartient à un arbre.

• Écrire une fonction qui retourne la liste résultant du parcours infixe d'un arbre : en chaque nœud, on parcourt le fils gauche, puis on note la valeur du nœud, puis on parcourt le fils droit.

• Écrire une fonction qui, étant donné un arbre de nombres A ajoute des feuilles à A. La valeur de chacune de ces nouvelles feuilles contient la somme des valeurs sur le chemin de la racine aux (anciennes) feuilles.

TD numéro 8
Arbres binaires de recherche

On considère maintenant des arbres dont les valeurs sont des nombres, et pour lesquels les nœuds sont ordonnés : en tout nœud, le sous-arbre gauche contient des nombres plus petits que celui du nœud, et le sous-arbre droit des nombres plus grands.

• Écrire une fonction booléenne qui teste qu'un arbre binaire est un arbre binaire de recherche.

• Écrire une fonction booléenne qui teste si une valeur appartient à un arbre binaire de recherche.

• Ecrire une fonction qui insère un nombre dans un arbre binaire de recherche.

• Ecrire une fonction qui trie une liste de nombres en passant par la construction d'un arbre binaire de recherche.

• Donner la spécification de la fonction mystère ci-dessous :

```
(define mystere
  (lambda (a x) ; a ABR, x nb
    (cond ((vide? a)'())
          ((= (valeur a) x) (list (valeur a)))
          ((< x (valeur a)) (cons (valeur a) (mystere (fils-g a) x)))
          (else (cons (valeur a) (mystere (fils-d a) x))))))
```