# NF16 - TP 4 – Arbres binaires de recherche

## Introduction

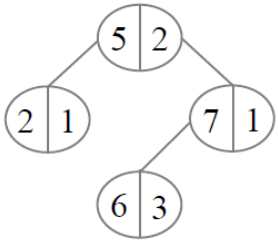
*Ce TP est basé sur le problème du final de NF16 proposé au Printemps 2015. Il a pour objectif de se familiariser avec les arbres binaires de recherche (ABR) et les différentes opérations nécessaires pour les manipuler.*

On souhaite gérer une liste d'entiers avec plusieurs occurrences. On utilise la notion d'arbre binaire de recherche (ABR), où on apporte une extension qui permet de gérer les occurrences d'un entier.

Un nœud x de l'arbre contient les informations suivantes :

- cle : une information de type entier, c'est la clé du nœud
- nb_occurrence : le nombre d'occurrences de 'cle' dans la liste d'entiers considérée
- gauche : un pointeur vers le fils gauche de x
- droit : un pointeur vers le fils droit de x

Exemple :
Soit la liste L= (5, 7, 6, 5, 2, 6, 6). Une représentation possible de cette liste L en ABR est la suivante :




Dans cette figure, la partie gauche d'un nœud représente sa clé alors que la partie droite représente le nombre d'occurrences de cette clé.

Remarque : *l'ABR ne doit pas comporter un nœud ayant le champ occurrence égal à 0.*

## A. Structures

1. Définir la structure **Noeud** qui représente un nœud de l'ABR et le type correspondant **T_Noeud**.
2. Définir la structure **Arbre** qui représente un ABR et le type correspondant **T_Arbre.**

## B. Fonctions requises

1. Implémentez la fonction **initABR** qui renvoie un pointeur vers un ABR vide.
2. Implémentez la fonction **creerNoeud** qui renvoie un pointeur vers un nœud dont la clé est l'entier entré en paramètre de la fonction.

3. Implémentez la fonction **ajouterElement** qui ajoute un entier donné en paramètre dans l'ABR.
4. Implémentez la fonction **afficherArbre** qui affiche tous les éléments de l'arbre triés en ordre croissant.
5. Implémentez la fonction **rechercherElement** qui vérifie si un élément existe dans l'ABR.
6. Implémentez la fonction **decrementerElement** qui permet de décrémenter le nombre d'occurrences dans l'ABR de la clé entrée en paramètre.

## C. Programme Principal :

Utiliser les fonctions précédentes pour proposer à l'utilisateur le menu interactif suivant :

1. Créer un ABR à partir d'une série de N entiers saisis par l'utilisateur
2. Créer un ABR à partir d'une série de N entiers lus depuis un fichier texte
3. Afficher tous les éléments de l'ABR
4. Rechercher un élément dans l'ABR
5. Supprimer une occurrence d'un élément de l'ABR
6. Quitter

## Consignes générales :

- **Sources**

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés. Vous devrez également être attentifs à la complexité des algorithmes implémentés.

L'organisation MINIMALE du projet est la suivante :

- Fichier d'en-tête tp4.h, contenant la déclaration des structures/fonctions de base,
- Fichier source tp4.c, contenant la définition de chaque fonction,
- Fichier source main.c, contenant le programme principal.

- **Rapport**

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous avez choisies d'implémenter et les raisons de ces choix.
- Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos fichiers source feront l'objet d'une remise de devoir sur Moodle dans l'espace qui sera ouvert à cet effet quelques jours suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).

**NB.** Les modalités de démonstration vous seront précisées en temps voulu.