Université du Québec **mercredi 25 mars 2015.**
à Chicoutimi (DIM)

Devoir 3
Programmation, Structure de données et algorithmes (8SIF109)
Ce devoir doit être remis au plus tard le vendredi 24 avril 2015 avant 16h.

Instructions

* Pour faciliter la correction de vos programmes, il est recommandé de bien les commenter.
* Envoyer à votre assistant votre devoir en incluant votre fichier source avec **noms et prénoms des co-équipiers** en tête de ce fichier ainsi que votre fichier exécutable.
* Travaillez en équipe au plus de deux étudiant(e)s, remettez une seule copie par équipe. Il est strictement interdit pour une équipe de copier le travail d'une autre équipe.
* **Attention:** ne pas modifier plus vos programmes après la date limite puisque cela va changer la date de la dernière modification de vos fichiers. Ne pas oublier pas non plus d'interdire l'accès à vos fichiers. Vous êtes le seul responsable de votre compte.

==========

Objectifs du travail demandé

Ce devoir consiste à implémenter et manipuler des arbres binaires de recherche avec des classes en C++. Les éléments contenus dans cet arbre sont tous des entiers. À l'aide d'un fichier d'entrée qui vous sera fourni ultérieurement, vous devez coder les fonctionnalités suivantes :

1. Construire un arbre binaire de recherche.
2. Détruire cet arbre binaire de recherche.
3. Insérer un élément dans cet arbre.
4. Supprimer un élément de cet arbre.
5. Afficher les éléments de cet arbre niveau par niveau (voir aussi le bonus).
6. Afficher la hauteur de cet arbre, en convenant qu'un nœud vide à une hauteur égale à -1.
7. Afficher les ascendants d'un élément donné.
8. Stocker dans un fichier cet arbre en utilisant l'implantation séquentielle. Noter l'utilisation du symbole / pour indiquer un lien nul.

Le fichier texte d'entrée contient une suite d'opérations à effectuer sur cet arbre :

I,d	pour insérer l'élément de valeur **d** dans l'arbre binaire de recherche.
S,d	pour supprimer l'élément de valeur **d** de l'arbre binaire de recherche.
A	pour afficher les éléments de l'arbre binaire de recherche niveau par niveau (voir aussi bonus).
H :	pour afficher la hauteur de l'arbre binaire de recherche.
G,d	pour afficher les ascendants du nœud de valeur **d**.
T	pour archiver l'arbre binaire de recherche dans un fichier texte en implémentation séquentielle.

Les structures de données à utiliser sont les suivantes:

```
struct noeud {
  int  valeur;
  noeud *gauche;
  noeud *droit;
};
class ABR {
private:
  noeud *racine;  // La racine de l'arbre binaire de recherche.
public:
 ABR (nœud *racine); // Construit l'arbre dont la racine est à l'adresse racine.
```

~ABR( ); // Désalloue l'espace mémoire occupé par l'arbre dont la racine est à l'adresse racine.
void Inserer(nœud *racine, int d); // insère le nœud de valeur **d** dans l'arbre.
void Supprimer(nœud *racine, int d) // Supprime le nœud de valeur **d** de l'arbre.
void Afficher_Arbre(nœud *racine); //Affiche les éléments de l'arbre niveau par niveau**. Un bonus de 5pts** est donné à ceux et celles qui commencent par le premier niveau, ensuite le deuxième niveau, ainsi de suite, jusqu'à arriver à la racine ; le premier niveau étant l'ensemble des nœuds qui n'ont pas de descendants.
int Afficher_hauteur(nœud *racine); //Affiche la hauteur de l'arbre.
void Afficher_Ascendant(nœud *racine, int d); // Affiche les ascendants du nœud de valeur **d**.
void Archiver (nœud *racine); // Archiver en implémentation séquentielle l'arbre dont la racine est racine dans un fichier texte que vous allez définir.
(**bonus 10pts**)
}; // fin de la déclaration de la classe ABR