Série de TP5 : Arbres

Exercice 1 : Arbre Binaire (AB)

On souhaite ranger une suite de nombres entiers dans un arbre binaire.
Dans un **arbre binaire (AB)** chaque nœud possède au plus deux nœuds fils appelés gauche et droit.

Définissez les opérations suivantes :

1. la structure d'un arbre binaire (AB).
2. Initialiser l'arbre.
3. Créer un élément.
4. Vérifier si un arbre est vide.
5. **Parcours de l'arbre :** afficher son contenu en utilisant les choix de parcours suivants :

Pré-ordre :
- Traiter la racine de l'arbre
- Traiter sa partie gauche
- Traiter sa partie droite

In-ordre :
- Traiter sa partie gauche
- Traiter la racine de l'arbre
- Traiter sa partie droite

Post-ordre :
- Traiter sa partie gauche
- Traiter sa partie droite
- Traiter la racine de l'arbre

6. Comptabiliser le nombre d'éléments d'un arbre.
7. Vérifier si un nœud est feuille ou non.
8. Afficher toutes les feuilles.
9. Calculer la hauteur d'un arbre.

Exercice 2 : Arbre Binaire de Recherche (ABR)

Un Arbre Binaire de Recherche (ABR) est construit de façon à ce qu'il soit facile de faire une recherche sur les valeurs qu'il contient. Pour cela, les deux règles suivantes sont définies :

- Toutes les valeurs inférieures ou égales à celle de la racine sont stockées dans le sous-arbre gauche.
- Toutes les valeurs strictement supérieures à celle de la racine sont stockées dans le sous-arbre droit.

- Les sous-arbres gauche et droit de la racine sont aussi des ABR.

- Définissez une structure **struct ABR** permettant de coder un nœud d'ABR contenant un entier.
- Ecrivez les fonctions suivantes :

1. **creer_arbre** qui prend en argument un entier et renvoie un ABR d'un seul nœud contenant cette valeur.
2. **afficher_croissant** permettant d'afficher les valeurs des nœuds d'un ABR de manière croissante.
3. **afficher_arbre** permettant d'afficher les valeurs des nœuds d'un ABR de manière à voir la structure de l'arbre. Un nœud sera affiché par (g,v,d) où g est le sous-arbre gauche, v la valeur du nœud et d le sous-arbre droit. Par exemple, l'arbre présenté dans la figure ci-dessous sera affiché par : (((_,1,_),3,_),4,((_,6,_),6,((_,7,_),9,_))). Les '_' indiquent les sous-arbres vides.
4. **inserer** qui ajoute une valeur dans l'ABR (ce sera un nouveau nœud placé correctement dans l'arbre).
5. **trouver_noeud** qui prend en argument un entier et un arbre et qui renvoie l'adresse d'un nœud de l'arbre contenant cet entier (ou NULL si cet entier n'est pas dans l'arbre).