Nom : _____

Prénom : _____

Numéro étudiant : | | | | | | | | |

Les Arbres Binaires de Recherche

4 décembre 2020

A lire absolument :

1. L'objectif n'est pas d'apprendre la correction par cœur, mais de comprendre les mécanismes mis en œuvre. Cela vous permettra de vous adapter face à un problème nouveau.
2. En particulier, vous devez être capable de refaire l'intégralité du sujet, seul, sans aucune aide ni support.
3. Votre travail sera corrigé automatiquement par l'outil de correction automatique CAT. Cela implique que vous devez respecter scrupuleusement les consignes de chaque exercice. Faites très attention aux messages qu'il vous est demandé d'afficher. Un espace en trop, un saut de ligne en moins et l'exercice risque d'échouer.
4. L'enseignant voit votre activité sur le site, ainsi que l'historique de vos dépôts. Pensez à déposer votre travail régulièrement afin qu'il puisse vous apporter des conseils personnalisés.
5. Si l'enseignant vous demande de rendre votre travail sur papier, vous devez répondre directement sur le sujet en respectant absolument la zone prévue à cet effet. Tout ce qui se trouve en dehors de la zone sera ignoré.
6. Si le sujet contient un QCM, vous devez colorier les cases avec un stylo bleu ou noir. Les autres couleurs seront ignorées.
7. Chaque feuille est identifiée de manière unique. Vous pouvez donc rendre votre sujet avec les feuilles mélangées, mais il est préférable de les trier car cela vous permet de vérifier que vous n'en avez pas oublié une.
8. Si vous faites face à un problème, un bug, une erreur ou que vous souhaitez participer à l'amélioration de la plateforme, envoyez un mail à l'adresse suivante : support-cat@liste.parisnanterre.fr

Ne rien écrire dans cette zone

Dans ce TD, nous allons implémenter un Arbre Binaire de Recherche (ou ABR) et des fonctions pour les manipuler. Un ABR est tout d'abord « binaire »ce qui implique que chacun de ses noeuds peut comporter au maximum deux fils. Il est aussi construit de façon à ce qu'il soit facile de faire une recherche sur les valeurs qu'il contient. Pour cela, on définit les règles suivantes :

- toutes les valeurs strictement inférieures à celle de la racine sont stockées dans le sous-arbre gauche.
- toutes les valeurs supérieures ou égales à celle de la racine sont stockées dans le sous-arbre droit.
- Cette définition est récursive, ce qui implique que les sous-arbres gauche et droit de la racine sont aussi des ABR.

Définition de la structure d'ABR

Définissez une structure (nommée struct arbre) permettant de coder un noeud d'ABR contenant un entier (nommé valeur). Votre structure devra contenir deux pointeurs vers des éléments de même type (nommés fils_gauche et fils_droit). Renommez ensuite votre nouveau type en ABR.

Création d'un noeud

Écrire une fonction qui prend en argument un entier et renvoie un ABR avec un seul élément contenant cette valeur.

```
1 ABR * creation_noeud(int v);
```

Ne rien écrire dans cette zone

Insertion d'un noeud dans un ABR

Écrire une fonction récursive qui accepte en argument un pointeur vers un arbre binaire de recherche ainsi qu'un noeud (un ABR ne contenant qu'un seul élément). Cette fonction doit ajouter ce noeud à l'ABR de telle sorte qu'il se trouve en feuille de l'arbre.

```
1 ABR * insertion(ABR * mon_arbre, ABR * noeud);
```

Ne rien écrire dans cette zone

Valeur minimum

Écrire une fonction récursive qui accepte en argument un pointeur vers un ABR contenant au moins un élément (pas besoin de vérifier) et qui retourne un pointeur vers le noeud possédant la valeur la plus faible.

```
1 ABR * get_min(ABR * r);
```

Valeur maximum

Écrire une fonction récursive qui accepte en argument un pointeur vers un ABR contenant au moins un élément (pas besoin de vérifier) et qui retourne un pointeur vers le noeud possédant la valeur la plus forte.

```
1 ABR * get_max(ABR * r);
```

Affichage d'un ABR

Écrire une fonction récursive qui affiche un ABR dans la console. Pour cela, vous devrez afficher le triplet "(SAG,val,SAD)". SAG représente l'affichage (récursif du sous arbre gauche, SAD celui du sous arbre droit, et val la valeur de la racine de l'arbre. Un ABR vide devra être représenté par la chaine "_".

```
1 void affichage(ABR * mon_arbre);
```

Ne rien écrire dans cette zone

Recherche d'une valeur

Écrire une fonction récursive qui accepte en argument un pointeur vers un arbre binaire de recherche ainsi qu'un entier v et qui retourne l'adresse du premier noeud contenant la valeur v. Si cette valeur n'est pas présente dans l'arbre, retourner NULL.

```
1 ABR * recherche(ABR * mon_arbre, int v);
```

Père d'un noeud

Écrire une fonction qui, à partir d'un ABR et d'un noeud n de cet ABR, va retourner l'adresse du noeud dont n est le successeur (fils gauche ou droit). Si n ne possède pas de père, alors retourner NULL ;

```
1 ABR * get_pere(ABR * r, ABR * f);
```