Nom : MOSAAD
Prénom : Chehab
Numéro étudiant : 22106126

TP6 : Arbres binaires de recherche équilibrés - Arbres Rouge-Noir.

1) **Description de l'archive logicielle fournie.**
2) **Propriétés des arbres Rouge-Noir.**
3) **Coloration et rotations :**
    1. **Exercice 1 : extension de la structure de données et coloration de l'arbre :**
        1) Le type NodeColor doit être déclaré dans bstree.c parce qu'il est utilisé uniquement dans le fichier bstree.c et n'est pas nécessaire dans les autres fichiers source ou les fichiers qui incluent bstree.h, il n'est pas nécessaire de l'ajouter dans bstree.h.

        2) La fonction void printNode(const BinarySearchTree *n, void *out) est chargée d'imprimer la représentation graphique du nœud de l'arbre. La première ligne de la fonction déclare un pointeur de fichier file et le cast du pointeur générique userData vers un pointeur de fichier FILE *. Ensuite, la fonction utilise la fonction bstree_color pour obtenir la couleur du nœud et la stocke dans une variable appelée color. La ligne suivante utilise la fonction fprintf pour écrire dans le fichier. Elle imprime une chaîne de caractères formatée qui représente le nœud, y compris son étiquette, ses liens vers le parent, le fils gauche et le fils droit, ainsi que sa couleur de remplissage. La couleur de remplissage dépend de la couleur du nœud (red si le nœud est rouge ou si le nœud est la racine et tous ses fils sont nuls, sinon gray).
        Ensuite, la fonction vérifie si le fils gauche du nœud existe (bstree_left(n)). Si c'est le cas, elle utilise fprintf pour écrire dans le fichier une ligne représentant le lien entre le nœud et son fils gauche. Sinon, elle imprime une représentation d'un nœud NIL (nœud nul) avec une couleur de remplissage grise. Elle écrit également une ligne représentant le lien entre le nœud et le nœud NIL fils gauche. La même logique est appliquée pour le fils droit du nœud.
        Doit être déclaré dans bstree.c parce que ce n'est pas nécessaire dans les autres fichiers source ou les fichiers qui incluent bstree.h donc il n'est pas nécessaire de l'ajouter dans bstree.h.

        3)
```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> make
"Generating in release mode"
make[1]: Entering directory 'D:/INFO PAUL SABATIER/L2/S4/algo3/tp6/Code'
make[1]: Leaving directory 'D:/INFO PAUL SABATIER/L2/S4/algo3/tp6/Code'
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfilesimple.txt
Adding values to the tree.
        4 6 7 5 2 3 1
Done.
Exporting the tree.

Done.
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile1.txt
Adding values to the tree.
        7 16 3 13 14 6 19 20 18 17 2 1 4 5 8 11 15 10 9 12
Done.
Exporting the tree.

Done.
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile2.txt
Adding values to the tree.
        1 2 3 4 5 6 7 8 9 10
Done.
Exporting the tree.

Done.
```

    2. **Exercice 2 : programmation des opérateurs de rotation :**
        1) La fonction void rightrotate(BinarySearchTree *y) effectue une rotation droite sur un nœud de l'arbre binaire de recherche. Elle prend en argument un pointeur vers un nœud t.

La première ligne de la fonction crée un pointeur l qui pointe vers le sous-arbre gauche de t. Ensuite, la fonction réarrange les liens entre les nœuds. Elle met à jour t->left pour qu'il pointe vers l->right, puis vérifie si t->left existe. Si c'est le cas, elle met à jour t->left->parent pour qu'il pointe vers t, indiquant ainsi le nouveau parent de t->left. Ensuite, la fonction met à jour l->parent pour qu'il pointe vers t->parent. Si t->parent est nul, cela signifie que t était la racine de l'arbre, donc root (la variable globale représentant la racine de l'arbre) est mise à jour pour pointer vers l. Sinon, la fonction vérifie si t était le fils gauche ou le fils droit de t->parent et met à jour t->parent->left ou t->parent->right en conséquence. Enfin, les liens entre l et t sont mis à jour pour effectuer la rotation droite. l->right est défini comme t, et t->parent est défini comme l.

La fonction void leftrotate(BinarySearchTree *x) est similaire à rightrotate, mais elle effectue une rotation gauche sur un nœud de l'arbre binaire de recherche. La première ligne de la fonction crée un pointeur r qui pointe vers le sous-arbre droit de t.
Ensuite, la fonction réarrange les liens entre les nœuds. Elle met à jour t->right pour qu'il pointe vers r->left, puis vérifie si t->right existe. Si c'est le cas, elle met à jour t->right->parent pour qu'il pointe vers t, indiquant ainsi le nouveau parent de t->right. Ensuite, la fonction met à jour r->parent pour qu'il pointe vers t->parent. Si t->parent est nul, cela signifie que t était la racine de l'arbre, donc root est mis à jour pour pointer vers r. Sinon, la fonction vérifie si t était le fils gauche ou le fils droit de t->parent et met à jour t->parent->left ou t->parent->right en conséquence. Enfin, les liens entre r et t sont mis à jour pour effectuer la rotation gauche. r->left est défini comme t, et t->parent est défini comme r.

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfilesimple.txt
Adding values to the tree.
        4 6 7 5 2 3 1
Done.
Exporting the tree.

Done.
Rotating the tree left around 6.
        Done.
Rotating the tree right around 6.

Done.
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile1.txt
Adding values to the tree.
        7 16 3 13 14 6 19 20 18 17 2 1 4 5 8 11 15 10 9 12
Done.
Exporting the tree.

Done.
Rotating the tree left around 14.
        Done.
Rotating the tree right around 14.

Done.
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile2.txt
Adding values to the tree.
        1 2 3 4 5 6 7 8 9 10
Done.
Exporting the tree.

Done.
Rotating the tree left around 4.
        Done.
Rotating the tree right around 4.

Done.
```

4) **Insertion d'une valeur dans un arbre rouge-noir :**
    1. **Restauration des propriétés des arbres Rouge-Noir après insertion.**
    2. **Exercice 3 : programmation des opérateurs de correction après insertion :**
        1) Dans l'opérateur ptrBinarySearchTree grandparent(ptrBinarySearchTree n) renvoie un pointeur vers le grand-parent de n s'il existe, sinon elle renvoie NULL. La fonction vérifie d'abord si n n'est pas nul et si le parent de n n'est pas nul. Si les deux conditions sont remplies, cela signifie que n a effectivement un parent et un grand-parent. Dans ce cas, la fonction renvoie le pointeur vers le grand-parent de n. Si l'une des conditions n'est pas

remplie (soit n est nul, soit le parent de n est nul), la fonction renvoie NULL, indiquant l'absence de grand-parent.

2) L'opérateur ptrBinarySearchTree uncle(ptrBinarySearchTree n) renvoie un pointeur vers l'oncle de n s'il existe, sinon elle renvoie NULL. La fonction utilise la fonction grandparent pour obtenir le grand-parent de n en appelant grandparent(n) et stocke le résultat dans une variable g. Ensuite, la fonction vérifie si g est nul. Si c'est le cas, cela signifie qu'il n'y a pas de grand-parent, ce qui implique qu'il n'y a pas d'oncle. Dans ce cas, la fonction renvoie NULL. Si g n'est pas nul, la fonction vérifie si le parent de n est le fils gauche de g. Si c'est le cas, cela signifie que l'oncle de n est le fils droit de g, donc la fonction renvoie le pointeur vers le fils droit de g. Si le parent de n n'est pas le fils gauche de g, cela signifie que le parent de n est le fils droit de g. Dans ce cas, l'oncle de n est le fils gauche de g, donc la fonction renvoie le pointeur vers le fils gauche de g.

3) L'opérateur ptrBinarySearchTree fixredblack_insert(ptrBinarySearchTree x) commence par appeler la fonction uncle(x) pour obtenir l'oncle de x et stocke le résultat dans une variable f. Ensuite, la fonction vérifie si f n'est pas nul et si la couleur de f est rouge. Si ces conditions sont remplies, cela signifie que l'oncle de x existe et qu'il est rouge. Dans ce cas, la fonction procède à une réorganisation de l'arbre pour résoudre le problème de non-consécution des nœuds rouges. Elle effectue les étapes suivantes :
Elle accède au grand-parent de x en appelant grandparent(x) et stocke le résultat dans une variable pp.
Elle change la couleur du parent de x en noir en définissant x->parent->color sur black.
Elle change la couleur de l'oncle de x en noir en définissant f->color sur black.
Elle change la couleur du grand-parent de x en rouge en définissant pp->color sur red.
Elle appelle récursivement la fonction fixredblack_insert(pp) pour continuer la résolution du problème à partir du grand-parent pp. Si les conditions initiales ne sont pas remplies (c'est-à-dire si l'oncle de x n'est pas rouge ou s'il est nul), la fonction passe à l'opérateur fixredblack_insert_case2 en appelant fixredblack_insert_case2(x).

4) L'opérateur ptrBinarySearchTree fixredblack_insert_case0(ptrBinarySearchTree x) commence par accéder au parent de x et le stocke dans une variable p. Ensuite, la fonction vérifie si le grand-parent de x est nul. Si c'est le cas, cela signifie que p est la racine de l'arbre, donc sa couleur est définie sur noir en définissant p->color sur black. Sinon, la fonction passe à l'opérateur fixredblack_insert_case1 en appelant fixredblack_insert_case1(x) pour résoudre davantage le problème de non-consécution des nœuds rouges. En fin de compte, la fonction renvoie x.

5) L'opérateur ptrBinarySearchTree fixredblack_insert_case1(ptrBinarySearchTree x) commence par vérifier si x->parent est nul. Si c'est le cas, cela signifie que x est la racine de l'arbre, donc sa couleur est définie sur noir en définissant x->color sur black. Sinon, la fonction vérifie si la couleur du parent de x est rouge. Si c'est le cas, cela indique une violation potentielle de la règle de non-consécution des nœuds rouges. Dans ce cas, la fonction passe à l'opérateur fixredblack_insert_case0 en appelant fixredblack_insert_case0(x) pour effectuer des ajustements supplémentaires. En fin de compte, la fonction renvoie x.

6) L'opérateur ptrBinarySearchTree fixredblack_insert_case2(ptrBinarySearchTree x) commence par déclarer des variables locales p et pp et leur assigne respectivement le parent et le grand-parent de x. Ensuite, elle utilise une condition pour vérifier si le parent p de x est le fils gauche du grand-parent pp. Si c'est le cas, cela signifie que x est également le fils gauche de p. Dans cette situation, la fonction appelle la fonction de correction

spécifique fixredblack_insert_case2_left avec x en tant qu'argument pour effectuer les ajustements nécessaires.
Si la condition n'est pas satisfaite, cela signifie que x est le fils droit de p. Dans ce cas, la fonction appelle la fonction de correction spécifique fixredblack_insert_case2_right avec x en tant qu'argument pour effectuer les ajustements appropriés.
Enfin, la fonction renvoie le nœud x, qui peut avoir été modifié par la fonction de correction spécifique.

L'opérateur ptrBinarySearchTree fixredblack_insert_case2_left(ptrBinarySearchTree x) commence par accéder au parent de x et le stocke dans une variable p. Ensuite, elle accède au grand-parent de x et le stocke dans une variable pp. La fonction vérifie ensuite si x est égal au fils gauche de p. Si c'est le cas, cela indique une configuration spécifique requérant une rotation vers la droite. Dans cette situation, la fonction effectue les étapes suivantes :
Elle appelle la fonction rightrotate(pp) pour effectuer une rotation vers la droite sur pp.
Elle change la couleur du nœud parent p en noir en définissant p->color sur black.
Elle change la couleur du grand-parent pp en rouge en définissant pp->color sur red.
Enfin, elle renvoie le pointeur vers le nœud parent p.
Si x n'est pas le fils gauche de p, cela indique une autre configuration spécifique nécessitant une rotation vers la gauche suivie d'une rotation vers la droite. Dans cette situation, la fonction effectue les étapes suivantes :
Elle appelle la fonction leftrotate(p) pour effectuer une rotation vers la gauche sur p.
Elle appelle la fonction rightrotate(pp) pour effectuer une rotation vers la droite sur pp.
Elle change la couleur du nœud x en noir en définissant x->color sur black.
Elle change la couleur du grand-parent pp en rouge en définissant pp->color sur red.
Enfin, elle renvoie le pointeur vers le nœud x.

De même, l'opérateur ptrBinarySearchTree fixredblack_insert_case2_right(ptrBinarySearchTree x), une fonction qui commence par accéder au parent de x et le stocke dans une variable p. Ensuite, elle accède au grand-parent de x et le stocke dans une variable pp.
La fonction vérifie ensuite si x est égal au fils droit de p. Si c'est le cas, cela indique une configuration spécifique nécessitant une rotation vers la gauche.
Dans cette situation, la fonction effectue les étapes suivantes :
Elle appelle la fonction leftrotate(pp) pour effectuer une rotation vers la gauche sur pp.
Elle change la couleur du nœud parent p en noir en définissant p->color sur black.
Elle change la couleur du grand-parent pp en rouge en définissant pp->color sur red.
Enfin, elle renvoie le pointeur vers le nœud parent p.
Si x n'est pas le fils droit de p, cela indique une autre configuration spécifique nécessitant une rotation vers la droite suivie d'une rotation vers la gauche.
Dans cette situation, la fonction effectue les étapes suivantes :
Elle appelle la fonction rightrotate(p) pour effectuer une rotation vers la droite sur p.
Elle appelle la fonction leftrotate(pp) pour effectuer une rotation vers la gauche sur pp.
Elle change la couleur du nœud x en noir en définissant x->color sur black.
Elle change la couleur du grand-parent pp en rouge en définissant pp->color sur red.
Enfin, elle renvoie le pointeur vers le nœud x.

3. **Exercice 4 : recherche de valeurs dans un arbre Rouge-Noir :**

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile1.txt
Adding values to the tree.
        7 16 3 13 14 6 19 20 18 17 2 1 4 5 8 11 15 10 9 12
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 5 in the tree : true
        Searching for value 14 in the tree : true
        Searching for value 11 in the tree : true
        Searching for value 23 in the tree : false
        Searching for value 71 in the tree : false
Done.
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile2.txt
Adding values to the tree.
        1 2 3 4 5 6 7 8 9 10
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 5 in the tree : true
        Searching for value 14 in the tree : false
        Searching for value 11 in the tree : false
        Searching for value 23 in the tree : false
        Searching for value 7 in the tree : true
Done.
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfilesimple.txt
Adding values to the tree.
        4 6 7 5 2 3 1
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 2 in the tree : true
        Searching for value 5 in the tree : true
        Searching for value 8 in the tree : false
        Searching for value 1 in the tree : true
Done.
```

5) **Suppression d'une valeur dans un arbre rouge-noir :**

1. **Restauration des propriétés des arbres Rouge-Noir après insertion.**
2. **Exercice 5 : programmation des opérateurs de correction après suppression :**

1) L'opérateur ptrBinarySearchTree fixredblack_remove(ptrBinarySearchTree p, ptrBinarySearchTree x) commence par vérifier si le nœud parent p est NULL. Si c'est le cas, cela signifie que le nœud x est la racine de l'arbre et il est noir. Dans cette situation, la fonction renvoie simplement le pointeur vers le nœud x, car le double noir disparaît. Si le nœud parent p n'est pas NULL, la fonction continue en prenant une décision en fonction de la couleur du frère (brother) du nœud x. Le frère b est déterminé en fonction de la position de x par rapport à p. Si la couleur de b est noire, la fonction appelle fixredblack_remove_case1 pour effectuer des ajustements supplémentaires. Si la couleur de b n'est pas noire (c'est-à-dire rouge), la fonction appelle fixredblack_remove_case2 pour effectuer des ajustements spécifiques.

2) L'opérateur ptrBinarySearchTree fixredblack_remove_case1_left(ptrBinarySearchTree p, ptrBinarySearchTree x) commence par accéder au frère b en tant que fils droit de p. Ensuite, elle vérifie si les deux fils de b sont noirs ou des feuilles nulles. Dans cette situation (cas 1.a), la fonction effectue les étapes suivantes :
   Elle change la couleur de b en rouge en définissant b->color sur red.
   Si la couleur de p est rouge, elle change la couleur de p en noir en définissant p->color sur black et renvoie p.
   Sinon, elle appelle fixredblack_remove avec les arguments p->parent (grand-parent de x) et p pour continuer les ajustements.
   Si le fils droit de b est rouge (cas 1.b), la fonction effectue les étapes suivantes :
   Elle appelle leftrotate(p) pour effectuer une rotation vers la gauche sur p.
   Elle attribue la couleur de p à b->color pour conserver la propriété des couleurs.
   Elle change la couleur de p en noir en définissant p->color sur black.
   Elle change la couleur du fils droit de b en noir en définissant b->right->color sur black.

Enfin, elle renvoie b.
Si le fils gauche de b est rouge (cas 1.c), la fonction effectue les étapes suivantes :
Elle change la couleur du fils gauche de b en noir en définissant b->left->color sur black.
Elle change la couleur de b en rouge en définissant b->color sur red.
Elle effectue une rotation vers la droite sur b en appelant rightrotate(b).
Elle effectue une rotation vers la gauche sur p en appelant leftrotate(p).
Elle attribue la couleur de p à b->color pour conserver la propriété des couleurs.
Elle ne modifie pas x.
Enfin, elle renvoie le pointeur vers le parent de b.

3) L'opérateur ptrBinarySearchTree fixredblack_remove_case1_right(ptrBinarySearchTree p, ptrBinarySearchTree x) fonctionne de manière similaire à fixredblack_remove_case1_left, mais il est utilisé lorsque le frère b est situé à droite de p. Les étapes et les ajustements effectués sont symétriques à ceux de fixredblack_remove_case1_left.

4) L'opérateur ptrBinarySearchTree fixredblack_remove_case1(ptrBinarySearchTree p, ptrBinarySearchTree x) est utilisé pour déterminer quelle fonction de correction spécifique doit être appelée en fonction de la position de x par rapport à p. Si x est le fils gauche de p, la fonction appelle fixredblack_remove_case1_left, sinon elle appelle fixredblack_remove_case1_right. La fonction renvoie le résultat de la fonction de correction spécifique.

5) L'opérateur ptrBinarySearchTree fixredblack_remove_case2_left(ptrBinarySearchTree p, ptrBinarySearchTree x) est utilisé lorsque le frère f du nœud x est situé à droite de p et est rouge. Il effectue les ajustements nécessaires pour rétablir les propriétés de l'arbre après la suppression.
La fonction commence par accéder au frère f en tant que fils droit de p. Ensuite, elle effectue les étapes suivantes :
Elle effectue une rotation vers la gauche sur p en appelant leftrotate(p).
Elle change la couleur de p en rouge en définissant p->color sur red.
Elle change la couleur de f en noir en définissant f->color sur black.
Elle appelle fixredblack_remove_case1_left pour effectuer d'autres ajustements spécifiques à gauche avec p et x en tant qu'arguments. Enfin, elle renvoie f.

L'opérateur ptrBinarySearchTree fixredblack_remove_case2_right(ptrBinarySearchTree p, ptrBinarySearchTree x) fonctionne de manière similaire à fixredblack_remove_case2_left, mais il est utilisé lorsque le frère f est situé à gauche de p. Les étapes et les ajustements effectués sont symétriques à ceux de fixredblack_remove_case2_left.

L'opérateur ptrBinarySearchTree fixredblack_remove_case2(ptrBinarySearchTree p, ptrBinarySearchTree x) est utilisé pour déterminer quelle fonction de correction spécifique doit être appelée en fonction de la position de x par rapport à p. Si x est le fils gauche de p, la fonction appelle fixredblack_remove_case2_left, sinon elle appelle fixredblack_remove_case2_right. La fonction renvoie le résultat de la fonction de correction spécifique.

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> make
"Generating in release mode"
make[1]: Entering directory 'D:/INFO PAUL SABATIER/L2/S4/algo3/tp6/Code'
make[1]: Leaving directory 'D:/INFO PAUL SABATIER/L2/S4/algo3/tp6/Code'
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfilesimple.txt
Adding values to the tree.
        4 6 7 5 2 3 1
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 2 in the tree : true
        Searching for value 5 in the tree : true
        Searching for value 8 in the tree : false
        Searching for value 1 in the tree : true
Done.
Removing from the tree.
        Removing the value 2 from the tree :
        Removing the value 5 from the tree :
        Removing the value 8 from the tree :
        Removing the value 3 from the tree :
        Removing the value 1 from the tree :
Done.
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code>
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile1.txt
Adding values to the tree.
        7 16 3 13 14 6 19 20 18 17 2 1 4 5 8 11 15 10 9 12
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 5 in the tree : true
        Searching for value 14 in the tree : true
        Searching for value 11 in the tree : true
        Searching for value 23 in the tree : false
        Searching for value 71 in the tree : false
Done.
Removing from the tree.
        Removing the value 17 from the tree :
        Removing the value 9 from the tree :
        Removing the value 14 from the tree :
        Removing the value 11 from the tree :
        Removing the value 7 from the tree :
Done.
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp6\Code> .\redblack_trees.exe ..\Test\testfile2.txt
Adding values to the tree.
        1 2 3 4 5 6 7 8 9 10
Done.
Exporting the tree.

Done.
Searching into the tree.
        Searching for value 5 in the tree : true
        Searching for value 14 in the tree : false
        Searching for value 11 in the tree : false
        Searching for value 23 in the tree : false
        Searching for value 7 in the tree : true
Done.
Removing from the tree.
        Removing the value 8 from the tree :
        Removing the value 9 from the tree :
        Removing the value 10 from the tree :
        Removing the value 5 from the tree :
        Removing the value 8 from the tree :
Done.
```