UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 339

Laboratoire#5 : Type `map`

Le but de ce laboratoire est de continuer à pratiquer l'implantation de TAD fonctionnellement équivalents à ceux de la SL. Vous devez compléter une classe `map` à peu près équivalente fonctionnellement à celle de la bibliothèque standard (SL) de C++.

Ce devoir est à faire en équipe de deux obligatoirement. Il devra être complété avant le vendredi 17 avril 2026 à 23h59. Vous devez remettre, sur `turnin.dinf.usherbrooke.ca`, les fichiers générés au cours de ce laboratoire.

Description de la tâche à réaliser

On vous fournit le code de base d'une classe générique `map`. La technique de représentation choisie est un `arbre AVL` avec des noeuds permettant de remonter vers le parent et descendre vers les enfants. Un noeud contient cinq éléments :

`CONTENU` : C'est un pointeur vers un élément de type `PAIRE` (constituée d'une clé constante et une valeur modifiable) stocké à ce noeud.

`PARENT, GAUCHE, DROITE` : Ce sont des pointeurs vers les noeuds parent et enfants du noeud.

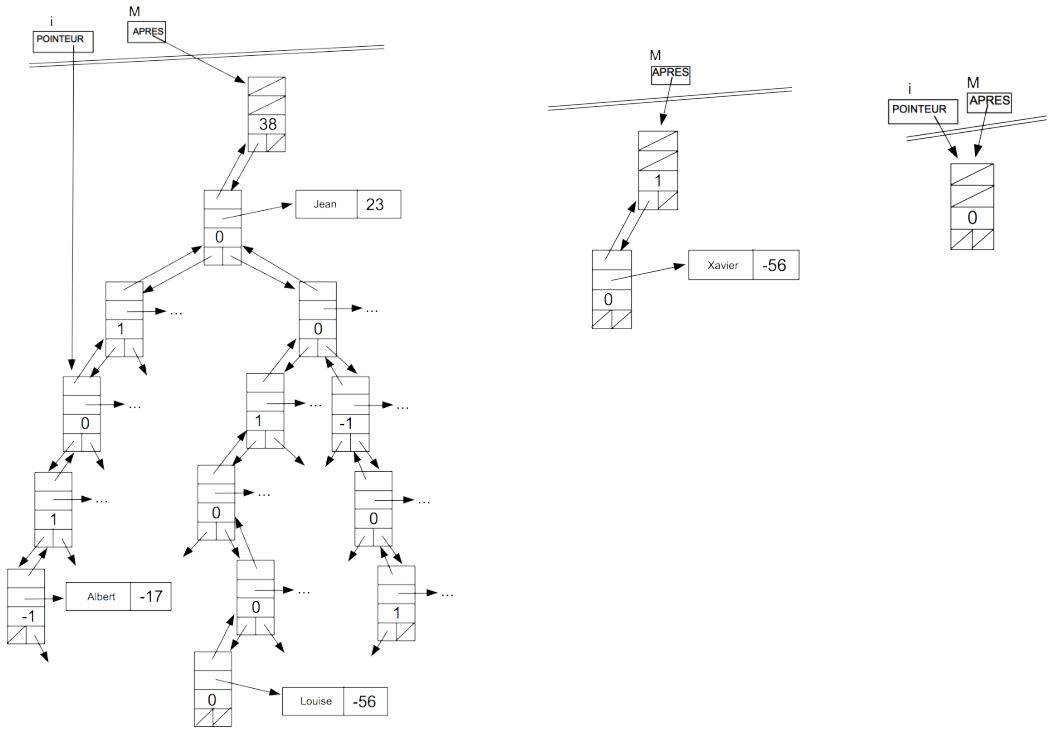
`INDICE` : C'est un entier, l'indice d'équilibre du noeud.

Dans la représentation choisie, on ajoute un noeud au dessus de la racine pour représenter la position de fin après le dernier élément. Ce noeud supplémentaire nous permet de stocker le nombre d'éléments que contient l'arbre au total dans son attribut `INDICE`. On voit ci-après l'illustration d'un `map` de 38 éléments, un autre de 1 élément, et un `map` vide. Un itérateur est représenté par un pointeur de noeud.

Les fonctions d'insertion et de suppression à partir d'une clé vous sont déjà données. Elles insèrent et enlèvent correctement les éléments, mais sans faire de rééquilibrage. Les indices d'équilibre sont cependant correctement calculés. Pour que les fonctions d'insertion et de suppression fonctionnent correctement, il reste à coder les fonctions de rotation vers la gauche ou vers la droite.

Les fonctions que vous avez à écrire sont :

`lower_bound` : Elle reçoit un objet du type de ceux stockés dans l'arbre, et retourne un itérateur donnant la position soit de cet objet s'il est présent, soit du premier objet qui lui est supérieur, soit de la fin si l'objet est supérieur à tous les éléments de l'arbre.



`insert` et `erase` à partir d'un iterateur : Elles réalisent l'insertion ou la suppression en temps constant amorti. Elles sont nécessaires pour que la classe soit admissible à faire partie de la bibliothèque normalisée.

`erase` à partir d'une clé : Elle retourne 1 si on a effectivement enlevé l'élément, 0 sinon.

`rotation_gauche_droite` et `rotation_droite_gauche` : Ce sont les deux fonctions de rotation, vers la gauche et vers la droite. Elles sont nécessaires pour que les fonctions d'insertion et de suppression fonctionnent correctement. On peut utiliser la fonction `std::max` pour calculer le maximum de deux ou plusieurs valeurs.

Utilisez la fonction `afficher` pour tester la conformité de votre code avec la spécification. Pour chaque noeud, elle affiche le contenu et l'indice. Elle essaie aussi de dessiner grossièrement la structure de l'arbre, sa partie droite étant en haut du dessin.

Vous avez aussi une fonction `verifier_hauteurs` qui vous permet d'explorer l'arbre pour déterminer si tous les noeuds sont correctement équilibrés. Elle retourne vrai si tout est OK. Sinon, elle affiche un noeud erroné et retourne faux.

Remise du travail

Pour soumettre votre travail, connectez-vous, dans un fureteur, au serveur `http://turnin.dinf.usherbrooke.ca` en utilisant votre CIP, puis choisissez le cours `IFT339` et le projet `TP5`. Chargez votre fichier `map2.h` et soumettez-le. Indiquez bien les NOMS (pas les matricules) des deux membres de l'équipe en commentaire dans ces fichiers. Ne faites qu'une seule soumission par équipe. Ne remettez pas d'autre fichier, ni d'exécutable.

Barême

- 25 points pour soumission réussie, par une équipe de deux, d'un programme qui compile sans erreur
- 10 points pour respect des normes de programmation (se référer au document sur les normes de programmation sur le site web du cours)
- 40 points pour le respect de la conception et des instructions fournies (spécifications des opérateurs et instructions de remise)
- 25 points la complétion correcte du code (5 points pour la fonction `insert` à partir d'un iterateur, et 4 points pour chacune des cinq autres fonctions).