C++ Implementation of AVL Trees

Mingyu Guo

1 Task Description

You are asked to use C++ to implement

- Binary Tree Traversal
- AVL Tree Insertion and Deletion

2 Submission Guideline

Your submission should contain exactly one file: `main.cpp`

There are be different versions of AVL trees. You should implement the one specified on the slides (e.g., when you delete a node with two children, swap the value with the largest value on the left).

You should start your program by initializing an empty AVL tree. Your program takes one line as input. The input line contains n "modification moves" separated by spaces ($1 \leq n \leq 100$). The available modification moves are

- `Aint` (Character `A` followed by an int value between 1 and 100): `A3` means insert value 3 into the AVL tree. If 3 is already in the tree, do nothing.
- `Dint` (Character `D` followed by an int value between 1 and 100): `D3` means delete value 3 into the AVL tree. If 3 is not in the tree, do nothing.

Your input is then followed by exactly one finishing move (`PRE` or `POST` or `IN`): If the finishing move is `PRE`, then you should print out the tree (in its current situation) in pre-order. If the tree is empty, print out `EMPTY`. Otherwise, print out the values separated by spaces. `POST` and `IN` are handled similarly.

You don't need to worry about invalid inputs.

Sample input 1: `A1 A2 A3 IN`
Sample output 1: `1 2 3`
Sample input 2: `A1 A2 A3 PRE`
Sample output 2: `2 1 3`
Sample input 3: `A1 D1 POST`
Sample output 3: `EMPTY`