CS151:Spring 2023

Lab 10

Write a `AVLTee` that extends your `LinkedBinaryTree` (from A6)

1. Add a `parent` reference and a `height` instance variable to the `Node` class of `LinkedBinaryTree`. Inheritance of inner/nested classes of Java could be weird so it's easier to just make changes to `Node` in `LinkedBinaryTree` instead.

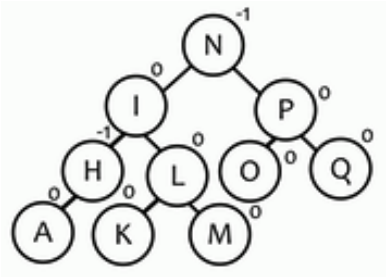
2. Modify/override `LinkedBinaryTree` and/or `AVLTree` so that `parent` and `height` are set correctly on insertion and deletion. You might need additional helper methods (to compute height, for example).

3. Override `toString` of the `Node` class to print the element followed by its height in parenthesis

4. Modify/Override `toStringInOrder` so that it uses the above `toString` of `Node` and returns a traversal string listing the elements with height attached.

5. Implement `rebalance` with associated helpers (`rotateLeft`, `rotateRight`, `rotateLeftRight`, `rotateRightLeft`) and call appropriately on insertion.

6. Create an `AVLTree<String>` and insert the exercise example given in class, i.e. "M", "N", "O", "L", "K", "Q", "P", "H", "I", "A" and the final balanced tree should look like this:



7. Override `toString` of `AVLTree` to print out the in-order traversal, with the height of each node attached. For example, the tree above should return the following string:

```
A(1) H(2) I(3) K(1) L(2) M(1) N(4) O(1) P(2) Q(1)
```