CSE 12 HW7 --- Binary Tree

General Notes

Overall this PA should be very straight forward. Make sure you understand the basic mechanism of the binary tree before you start the PA. If so, you should find the development process very easy and smooth.

Driver.java

Similar to hw6, you need to fill in all the methods that `UCSDStudent` need based on the `Variable` class of `Calculator.java`, which you can found on the course website code page.

Tree.java

Tree methods

- `debugOn()`
- `debugOff()`
- `Tree()`
  - The tree constructor
  - Just correctly assign the initial value of each data field.
  - Don't worry about the representation string.
- `jettison()`
  - Jettison the tree.
  - You need to jettison each node that the tree is currently holding.
  - You should call the `jettisonAllNodes` method.
  - After you jettison all the nodes, don't forget to jettison the tree itself.
- `jettisonAllNodes()`
  - This should be a recursive method, and you should utilize one of the tree traversal.
  - What traversal should you be using?
    - Post-order
    - Pre-order
    - In-order
  - If you need a hint to write this method, check out the `writeAllTnodes` method that we wrote for you.
    - What is the type of traversal for writing all the nodes?
    - How the two traversals differ in their implementation?
- `insert`
  - The insertion need to be *loop-based*.
  - You need to handle the two cases separately, an empty tree vs a tree has already has element in it.

        - You will be calling the constructors of the `TNode` and it requires you to pass in a `String` parameter in addition to the data.
            - This string should be `Tree.insert`
    - General case: when the tree is not empty, you will perform the following two steps. Both should be accomplished using loop.
        - Locating where to insert
            - Using a loop to iterate through your tree.
            - Make use of `isLessThan` and `equals` methods to perform comparisons and determine where to go down the tree.
                - Duplicate insertion need to be considered
                    - Do not forget to `jettison` the old data
                - When you can't go down the tree, meaning you shall place the element there
                    - Don't forget to set the parent after you place a new node.
        - Go up the tree to update the `height` and `balance` of each node.
            - When to terminate your loop?
                - What is the thing that is special for the `root`?
- `isEmpty`
    - Simple
- `lookup`
    - You should follow the exact procedure that you performed in insert for finding the node
    - When you reach the terminating conditions in `insert`, you should determine whether you found the item you are searching for, or the item is not in the tree.
    - You don't need to update `height` and `balance`.
- `remove`
    - This is lazy remove.
    - Very very similar to `lookup`.
    - If you found the item, then just mark it as being removed and decrement the `occupancy`.
    - But no need to update the `height` nor `balance`.
    - No need to jettison Node because it is lazy removal.

TNode methods

- `TNode()`
    - You should correctly initialize each data field.
        - What should the `left` and `right` be?
        - What should the `parent` be by default?
- `jettison`
    - Jettison the data and then the node itself.