CS10C SI Activity: Binary Search Tree

Question 1 (Binary Search) *Quick! A BST is best for which algorithm????? YES! That's correct, a binary search algorithm! You're a genius. Just knowing that already makes you 50% an expert with Binary Search Trees. Now, for the actual activity.*

*You will be programming a Binary Search Tree. No way!? Coding in a cs class? That's insane... is what you might say in the future *wink wink*. Since you'll be given a good portion of session to do this activity, it will be slightly longer/harder.* ***I suggest working together on this, create a repl.it!***

Here's the specs:

1. Create a the basic structure of a BST for integers (The tree class and a node class)
2. Create a **member** function that will insert a value we pass in
3. Create a function that will get an input from the user to insert into the tree
4. Create a **member** function that will remove a value we pass in
5. Create a function that will get an input from the user to remove from the tree
6. Create a **member** function that will search for a value we pass in
7. Create a function that will get an input from the user to search in the tree
   For the above functions, the run-time should be $\Theta(\lg n)$
8. Create a function that will print all values **inorder**
9. Test the print with input order: 5 1 -1 2 7 9 8 10 6

Question 2 (The Next Step) *That wasn't that bad... was it? I hope not. If it was difficult, was it at least easy to code the search part? The search function is one of the biggest part of a BST, so if you're confused about that part still, ask for help from your peers or your SI Leader.*

This next part might be a little harder, so if you need help, don't be afraid to ask. For this next part, you'll be making the BST into a more general BST for different data types. What does this mean? That's right! TEMPLATES! WOOHOOO. It's all coming back again. Of course, it's not just going to be templates, there'll be more additions. Well, it's time to code.

Here's the specs

1. Turn your frown upside down by making your BST into a template BST
   Hint: Both the node and tree classes will be changed.
2. Add a new data value within the tree that stores the total number of nodes within the tree
3. Add two print functions for printing **preorder AND postorder** (create three different prints)
   Hint: preorder=root,left,right. postorder=left,right,root
4. Add a new data value within the tree that stores the height of the tree
   Hint: Finding this value should still be $\Theta(\lg n)$
5. Add a new data value within the tree that stores the **SHORTEST** height in the tree
   Hint: Same hint as above
6. Output whether the tree is balanced or not (more specifically, if the difference between the total height and the shortest height is unbalanced if it's absolute value is more than 1)