Project 6: `BST-Roster ADT`

The Problem:

For this project you will **design and implement** a new ADT: **Roster**.
The Roster ADT is implemented as a **BST** that stores Student objects (as per the Student class in Project 1).

BST property: Roster operations are guaranteed to retain BST property, s.t. given any node, all Students in its left subtree are < the student at that node and all Students in its right subtree are >, where < and > are defined by last name alphabetical order as primary key and first name alphabetical order as secondary key (i.e., if the primary key is equal, compare the secondary key. No duplicates will be added to the roster - assume no two students will have same first and last names).

Implementation Requirements:

Your Roster ADT must implement at least the following methods:

- **Default constructor**
- **Copy constructor**
- **Destructor**
- bool **`isEmpty()`** //
- void **`add(Student)`** // adds the Student argument object maintaining the BST property described above
- void **`add(vector<Student>)`** // adds all the Student objects found in the vector argument, maintaining the BST property described above
- void **`remove(Student)`** //
- int **`getHeight()`** // as discussed in lecture, height is measured as the number of nodes on the longest path from root to leaf

- void **`display()`** // print all students inorder (as per inorder traversal), one per line, separated by comma (i.e. **`"id, first_name, last_name\n"`**)

Note: You MUST implement the Roster ADT using the BinaryNode class provided on Blackboard, not STL containers. **I reserve the prerogative to detract points given to your submission by Gradescope if you do otherwise.**
Also keep in mind that **`getHeight`** and **`display`** will be used to test other operations, so make sure they work correctly.

Extra Credit:

A **tree rotation** is an operation on a binary tree that changes the structure without interfering with the order of the elements. In a BST a rotation retains the BST property.
For a good description of rotating binary trees, see Wiki page.
Implement:

- void **`rotateLeft()`**
- void **`rotateRight()`**

Provided files (on Blackboard under Course Material/Project6):

- `BinaryNode` class
- `CourseMember` and `Student` class (you may assume overloaded operator ==)
- `Visitor` and `Printer` classes as discussed in lecture (usage is optional, but you may want to use it if you want to implement a general traversal and use the Printer functor to display the tree).

Testing:

Write a test driver (main function - not for submission) to incrementally test all methods in your ADT. You should instantiate Student objects to populate the Roster and test that all operations retain the BST property.

Submission:

You must **submit `Roster.hpp` and `Roster.cpp` (2 files)**

Your project must be submitted on Gradescope. The due date is Friday May 17 by 6pm. No late submissions will be accepted.

Have Fun!!!!!