1. Write a function that takes a BST of numbers and returns the sum of all the numbers in the tree.

2. Write a function that takes a BST of numbers and a minimum value as a parameter and removes all numbers less than the minimum value from the tree.

3. Write a function that takes a BST of numbers as a parameter and negates all the values in the tree. To keep the tree a legal BST you'll have to reverse the tree -- so left children become right children and vice versa.

4. Write a function that takes three parameters: a BST of numbers, a lower bound, and an upper bound. The function must return a sorted list of all the numbers in the BST which are greater than or equal to the lower bound and less than or equal to the upper bound.

5. The delete function we wrote in class replaces a node with two children with the maximum value in the left subtree. Write an alternate version of delete that replaces a node with the minimum value in the right subtree.

6. Using a single delete function (the one we wrote in class or the one from the preceding problem) will help un-balance a tree if used repeatedly. Write another version of delete which randomly decides between the two strategies (using the maximum of the left subtree or the minimum of the right subtree). This will help a tree remain more balanced after multiple deletes.