Binary Search Trees

In this lab exercise, you will be completing a partial implementation of a `BinarySearchTree` class for integer numbers.
Download the `BinarySearchTree.java` file and `Lab6Driver.java` files from C4, and add them to a new Java project.

The driver contains several test cases for insertion and removal. BSTTest0() generates random numbers and inserts them into a `BinarySearchTree` object. The test cases BSTTest1(),…, BSTTest8() test specific situations of removing keys from a BST. You can run each test case separately.

Methods to complete

`CalculateHeight` – this method recursively calculates the height of the (sub)tree rooted at its parameter. Hints:

- How is the height of a node related to the height of its subtrees?
- `Math.max(a,b)` returns the maximum of a and b.

`Insert` – inserts an item into the `BinarySearchTree` object. This can be completed iteratively in the public method `Insert(int item),` or handed off to a recursive private method `Insert(BSTNode nd, int item)`. Duplicate keys will be allowed and should go into the left subtree with the rule $left \leq current < right$.

`Remove` – Removes the specified key from the `BinarySearchTree` object, using the proper (non-lazy) method described in the lecture slides. The method returns false if the key is not found. Complete the code for the recursive method `Remove(BSTNode nd, int item)` that is called by the Remove method. Use the **predecesssor** replacement rule for the 2-child case. (for duplicate keys in the tree: only remove one key)

For all methods, pay special attention to special cases – empty tree, tree contains one element, etc.

Deliverables

Submit to C4:

- Your `BinarySearchTree.java` file with the above methods completed.