Assignment 6. **Trees.**

Given March 4, due Monday, March 23 (March 16 is an NYU holiday).

1. We start with a binary search tree, T, and delete x then y from T, giving a tree, T'. Suppose instead, we delete first y then x from T. Do we get the same tree T'. Give a proof or a counterexample.

2. Suppose T is a B-tree with $t = 3$, and suppose that there is no element in the tree with key x. If we insert x (more properly, an element with key x) into T and then delete x, do we get T back? Give a proof or a counterexample.

3. Give pseudocode (or actual code) that prints the keys from a binary search tree using a stack but not recursive function calls. This is in fact what happens in the computer, as recursion is implemented by stacks.

4. Find an algorithm that converts a complete binary search tree into a B-tree with three keys in every node and branching factor 4. Assume that the height of the binary tree is even. The algorithm should take $O(n)$ operations, where n is the number of keys in the tree. (Harder) Do the same without the assumption that the binary tree is balanced.

5. Write pseudocode that finds the successor to a given key in a B-tree.

6. I have a computer with a hard disk. The seek time is 500 times longer than the time needed to read a key, once the read head is in the right place. That is, the computer can read 500 successive keys in the time it takes to find the first one. The processor is much faster than either of these operations. What branching factor minimizes the time needed to get to a leaf in a balanced tree?