CSE 331 Algorithms and Data Structures
Homework 3 (Due at start of class on **2/15/2017**)

2/8/2017

1) Show that the maximum number of nodes in a binary tree of height h is 2^{h+1}-1. (Hint: try using induction. If you use induction, remember to include the base case in your proof. Note that other proof techniques can be used as well.) (6 points)

2) a. Show the result (after each insertion) of inserting 3, 1, 4, 6, 9, 2, 5, 7 into an initially empty binary search tree. (8 points)
   b. Show the result of deleting the root. (2 points)

3) Show the result of inserting 2, 1, 4, 5, 9, 3, 6, 7 into an initially empty AVL tree. Show the tree after each insertion and rotation. (11 points)

4) Suppose we want to add the operation `findKth` to our repertoire. The operation `findKth(k)` returns the kth smallest item in the tree. Assume all items are distinct. Explain how to **modify the binary search tree** to support this operation in O(log N) average time, without sacrificing the time bounds of any other operation. (5 points). (Using in-order traversal won't work because it does not support findKth(k) with O(logN) running time.)

5) What is the minimum number of nodes in an AVL tree of height 0? What is the minimum number of nodes in AVL trees of heights 1, 2, and 3? Let N(h) be the minimum number of nodes in an AVL tree with height h. Define a recursive function for N(h). (6 points)

6) Professor W thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor W claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a smallest possible counterexample to the professor's claim. (4 points)
   Note that the union of A and C cannot be empty. i.e. A or C must have some nodes.

7) Given a node in a BST, it is sometimes important to be able to find its successor in the sorted order determined by an in-order tree walk. If all keys are distinct, the successor of a node x is the node with the smallest key greater than key[x]. The structure of a BST allows us to determine the successor of a node easily. Describe a procedure that returns the successor of a node x in a binary search tree if it exists, and NIL if x has the largest key in the tree. (Hint: in order to consider all cases, find each node's successor in the tree shown on the last page. Then generalize your observations.) (8 points)
   Note that you **are not allowed to do** an in-order traversal to get successor of a node x.

8). Programming problem. (30 pts)

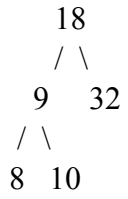
For a given file containing integer keys, construct a BST by **inserting each key into an initially empty tree using the given order**. Then print out the pre-order, in-order, and post-order traversal of the tree. The specific instructions are given below.

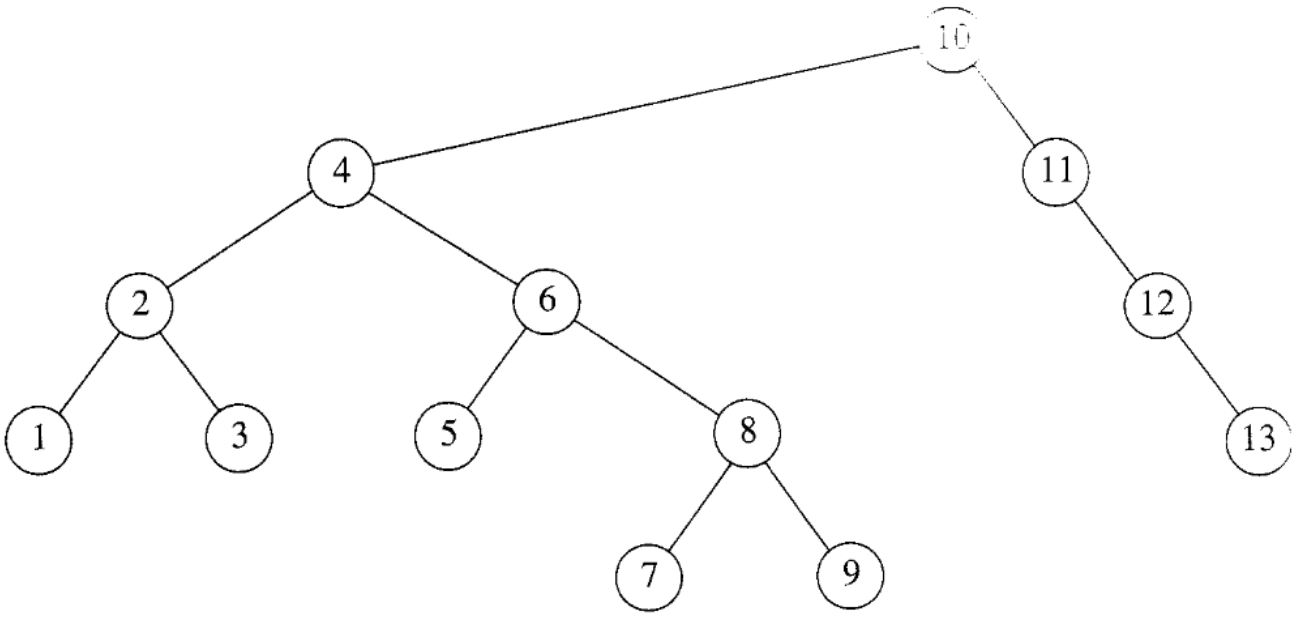
a. Your program must **take a file as input**. The input file is a list of integers separated by space (e.g. 18 9 32 8 3 4 0). This file is the **sole input** to your program. **Name your program as BST.(c/cpp/ java/pl etc.)**. We will compile it into BST (if there is a need to compile) and run it using the command: BST inputfile. Input file "inputfile" may be saved anywhere and can have different names. For example, your program should be able to accept the following commands:
"BST ../Section002/examples/testfile1.txt" or "BST ./testfiles/input1.txt" etc.

b. You can use any programming language. But it must be able to compile/run at arctic.cse.msu.edu. Any codes that we cannot compile or run get 0 point. Submit the source codes and a simple description about compiling your codes. Feel free to submit your own test input files. Submit them to Handin. No codes will be accepted via emails.

c. The output should be like this:
Pre-order traversal: x x x x x
In-order traversal: x x x x x (note: if this list is not sorted, your codes have bugs)
Post-order traversal: x x x x x

d. For example, if the input file is: 18 9 8 32 8 10, the final BST by calling insert function is:




10
4
11
2
6
12
1
3
5
8
13
7
9