Weekly Assignment 11:
Binary Search Trees

1. Draw the binary search tree that results when you insert items with keys

   ```
   9 5 12 7 4 10 8 11 18
   ```

   in that order into an initially empty tree. Remove 9 and 5 and draw the resulting trees after each removal.

2. Describe an $\mathcal{O}(n)$ algorithm for finding the second largest key in a binary search tree with n nodes. Include a complexity analysis.

3. Professor Balthazaar believes he has discovered an interesting property regarding Binary Search Trees (BST). Given a BST T and a search path for a key n where n is a leaf of the tree, we divide the tree into three sets:

   - L: The key values to the left of the search path,
   - P: The key values on the search path,
   - R: The key values to the right of the search path.

   The professor claims that for any $\ell \in L, p \in P, r \in R$ the following inequality holds $\ell \leq p \leq r$. Prove that the claim holds or give the smallest counterexample.

4. Recall that an AVL tree is a binary search tree in which, for every node, the height of both subtrees differs at most by one.
   (a) How many AVL trees can you build with the following elements: $\{1, 2, 3, 4, 5\}$? Draw them.
   (b) Write an $\mathcal{O}(n)$ time algorithm which returns a Boolean representing whether a given binary search tree with n elements is an AVL tree or not, that is, whether the height of the two subtrees of every node never differ by more than 1. Explain your answer, and argue why your solution is in $\mathcal{O}(n)$.

5. Give an $\mathcal{O}(n \lg n)$ algorithm which takes an integer array with n elements, and turns this into a balanced binary search tree (that is, with height $\mathcal{O}(\lg n)$. Explain your answer and include a complexity argument.