1. Suppose $S$ and $T$ are two ordered dictionaries each containing $n$ items. Both are implemented via sorted arrays and do not have any keys in common. Describe an $O(\log n)$-time algorithm that outputs the item whose key is the lower median in the union of $S$ and $T$. For example, when the keys stored in $S$ and $T$ are $S = [3, 6, 7, 9]$ and $T = [-1, 1, 2, 8]$, the lower median key is 3.

2. C-3.3. For this problem, your goal is to design an algorithm for $\text{findAllElements}(k)$ in a binary search tree in $O(h + s)$ time, where $h$ is the height of the tree and $s$ is the size of the output. This running time may look odd; if you think about it though not only should the running time depend on the height of the tree but also on how many items are in the output. If the output is large – say it’s all the items – then the dominant term in $h + s$ is $s$. But if the output is small, then the dominant term in $h + s$ is $h$. But because we don’t know in advance what the output will be, the running time is expressed as $O(h + s)$.

Note that your algorithm should work even if no item has key value equal to $k$.

3. Let $T$ be a binary search tree. Suppose we want to include a field $v$.size, the number of items stored in the subtree of $v$, at each node $v$.

a. Given $T$, describe an algorithm that will populate all the size fields in $T$. What is the running time of your algorithm?

b. Now suppose we make modifications to $T$ by inserting or removing items. For each modification – i.e., an insertion or a removal – describe the modifications that have to be made to the size fields of the nodes in $T$. How much time will such modifications take?

4. Pretty pictures. Your book does a really good job of illustrating binary trees. Here’s why:

(i) Nodes that have the same depth lie on the same horizontal line.

(ii) For an arbitrary node $v$, nodes that are in its right subtree lie to the lower right of $v$ while those in its left subtree lie to the lower left of $v$.

(See Figure 2.25 for example. In Figure 3.7 or 3.8, this is not exactly the case due likely to layout issues.)

Let $T$ be a binary tree with $n$ nodes. Suppose you are allowed to do some traversals on $T$ and store some values on the nodes (e.g., their depth, etc.) Describe an algorithm for drawing $T$ on an $n \times n$ grid. That is, for each node $v$ of $T$, assign it
integer coordinates \((i,j)\) so that when the nodes are drawn on their corresponding coordinates, the resulting drawing has two properties described above. (Assume that you are allowed to just say “Draw a line segment from \((i,j)\) to \((i',j')\).”) What is the running time of your algorithm?