1. Determine the order of the following list after each iteration of quicksort $[23, 20, 6, 17, 13, 25, 14]$, assume you have a way to take the median of the list as the pivot in each iteration.

The answer here will only include all the details for the first partition (the two subsequent iterations will be on arrays of size three, and hence are trivial).
We are assuming we can pick the median of the list, 17, in this case as the pivot.
First thing we do is swap the pivot with the first element in the list, getting:
$[17, 20, 6, 23, 13, 25, 14]$.
Next, we have the two pointers L and R starting on each end of the list and looking respectively for bigger and smaller elements with respect to the pivot. L will go to 20 and R will go to 14 in the first iterations. Swapping these elements results in:
$[17, 14, 6, 23, 13, 25, 20]$.
Continuing this process, we get L pointing to 23 and R pointing to 13, and swapping these results in:
$[17, 14, 6, 13, 23, 25, 20]$.
Now, R will go to 13 and will cross the L pointer, so this is when we swap the index with the R pivot, resulting in:
$13, 14, 6, 17, 23, 25, 20$. At this point, all elements smaller than the pivot are on its left and all elements bigger than the pivot are on its right so the partition is done.

2. Convert the array $a = [10, 24, 42, 66, 33, 9, 2, 30, 80, 48]$ into a minimum heap using the linear time heap building algorithm. Show the order of all items after each iteration.

Recall that the algorithm is performed recursively on the left sub-tree and then on the right sub-tree. But because this is a recursive call, it will keep on recursing on the left sub-tree of the...of the left sub-tree until it comes to a leaf, and only then will it recurses on the right sub-tree of the parent of the leaf. At any level of the recursion, when both the recursive calls are done, it will compare the parent with its two children (which, by virtue of the correctness of the algorithm are roots of minimum heaps), and if any one of the children has a key smaller than the root, it will swap the smallest of these with the key at the root node. We observe the step-by-step effects of this algorithm on the array below:
$a = [10, 24, 42, 66, 33, 9, 2, 30, 80, 48]$
$a = [10, 24, 42, \mathbf{30}, 33, 9, 2, \mathbf{66}, 80, 48]$
$a = [10, 24, \mathbf{2}, 30, 33, 9, \mathbf{42}, 66, 80, 48]$
$a = [\mathbf{2}, 24, \mathbf{10}, 30, 33, 9, 42, 66, 80, 48]$

3. Consider the following maximum heap: 39, 20, 37, 18, 6, 32, 13, 3, 14

   a) Insert 25 into the heap.

   b) Delete 39 from the heap.

   a) We always insert new nodes into a heap at the next available index so the list will be 39, 20, 37, 18, 6, 32, 13, 3, 14, 25 after the insertions. We then bubble the new node up to the correct place it should be and we get 39, 25, 37, 18, 20, 32, 13, 3, 14, 6.

   b) When removing a node, we always replace it with the last leaf. So we get 6, 25, 37, 18, 20, 32, 13, 3, 14 after removing 39. We then percolate the new root down to the correct place place it should be and get 37, 25, 32, 18, 20, 6, 13, 3, 14.

4. Prof X claims to have invented a sorting algorithm that sorts n items in time $\Theta(n)$.

   A. It is not in-place

   B. It is not comparison-based

   C. It is only doing it on average, not in the worst case

   D. It is not stable

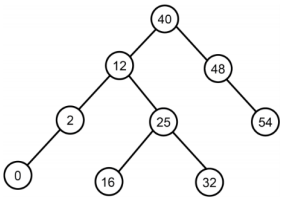
   E. Prof X is wrong

   B. The key notions to keep in mind here are, it is hard to disprove the existence of such an algorithm, especially since non-standard forms of computation exist, such as quantum comping and biological computing. However, as proven in class, a comparison-based algorithm cannot be sort an arbitrary array in time $\Theta(n)$ by virtue of it having to compare elements of the array.

5. Assuming that all keys involved are different, a max-heap that is also a binary search tree must consist of (choose the most accurate answer):
   A. at most 2 nodes
   B. a root node, or be empty
   C. a root node
   D. a root node, or root with one child
   E. a root node, or root with left child, or be empty

E is the answer. Even though other options, such as A, aren't false propositions, E is the most accurate answer in the sense of it explaining explicitly the only three possible cases such.

6. Given the binary search tree below:



   a) Delete node 48 in the tree.
   b) Delete node 16 in the tree.
   c) Delete node 12 in the tree by using the minimum key in the right subtree.
   d) Delete node 12 in the tree by using the maximum key in the left subtree.

a) Node 48 has only one child: delete the node, connect its child to its parent.

b) Node 16 has no children: simply delete.

c) Node 12 has two children: find the minimum key K = 16 in the right subtree, delete that node, and replace the key of node 12 by K.

d) Node 12 has two children: find the maximum key K = 2 in the left subtree, delete that node, and replace the key of node 12 by K.

7. Professor Bunyan 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 Bunyan claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Is this true? If so, explain how this is the case. If not, give a possible counterexample to the professor's claim.

This is not true, see counterexample. $A = \{9\}$, $B = \{8, 12, 10, 11\}$, and $C = \{13\}$