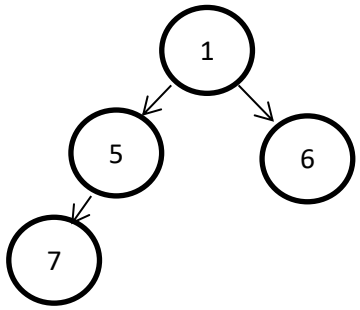
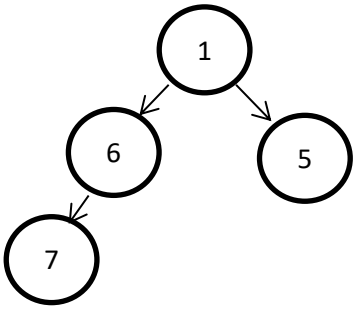
Question 1:

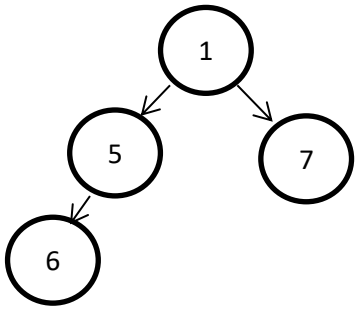
a) Heap 1:



Heap 2:

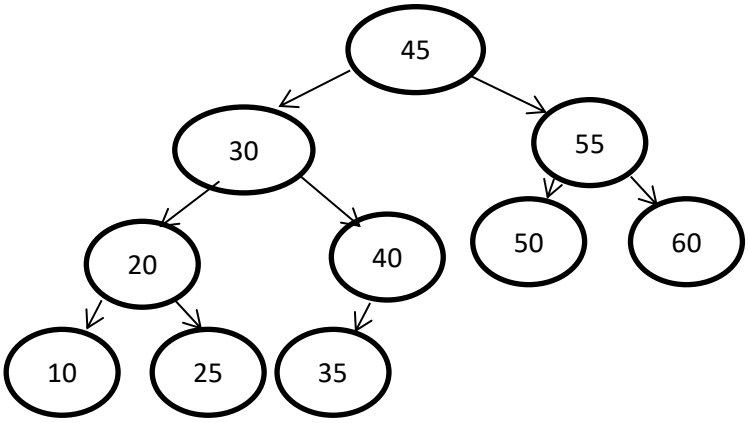


Heap 3:

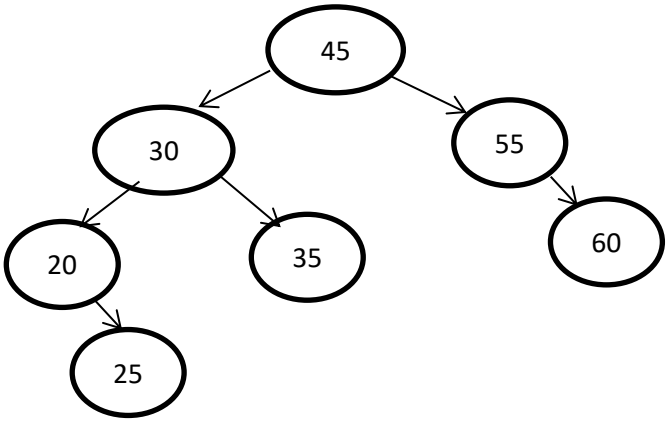


b)

Insert 40; 50; 45; 30; 60; 55; 20; 35; 10; 25 to empty AVL tree:



Delete 10 40 50 from the Tree :



Question 2:

Data Structure	Insert	extractMin
unsorted array	O (1)	O (n)
AVL tree	O(log n)	O (log n)
min-heap	O (log n)	O (log n) (heapRebuild after extraction, if deletion excluded, then, O (1))
unsorted linked list	O (1)	O (n)
sorted linked list	O (n)	O (1)

Question 4:

a) Expected running times for getLessThan and getGreaterThan methods are both O (n). Since the algorithm requires preorder traversal, and since preorder traversal requires O (n) running time, the running time for getLessThan and getGreaterThan also has O (n) running time.

Also, time complexity of getLessThan and getGreaterThan cannot be enhanced. It is impossible to find a better execution time than O (n) since we need to check all the items.

b) In question 3c, to find the median, one min-heap and one max-heap are used. getMedian method only gets the root of one of those heaps, the one that has more values, if both heaps have the same amount of values, then the average of the roots of heaps outputted. Also, **it requires O (1) time** because getting root value from a heap requires O (1) time.

This works because the actual job is done while inserting the items. Every item we add changes the current median value. An item, that is about to be inserted, is inserted to min-heap if the value of the item is greater than the current median. And the item is inserted to max-heap if the value is smaller than the current median. The maximum difference between heap sizes is always 1. The algorithm removes an item from one heap and adds to the other if size difference is more than 1. This is because we want to keep the median value in the middle. **This process requires O (log n) time** because inserting an item to a heap requires O (log n) time and after inserting, the rest of the algorithm also requires O (log n) time (if size difference > 1, remove and add).