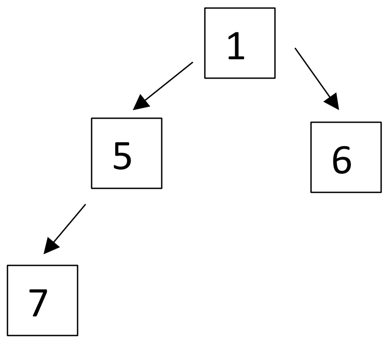
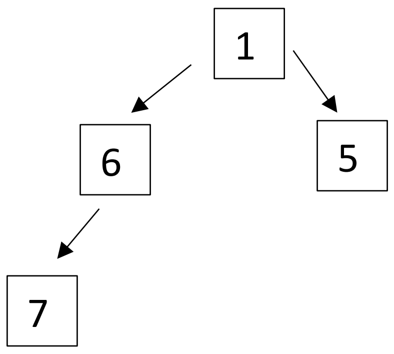
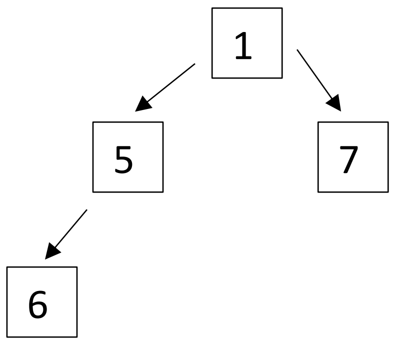
Question-1

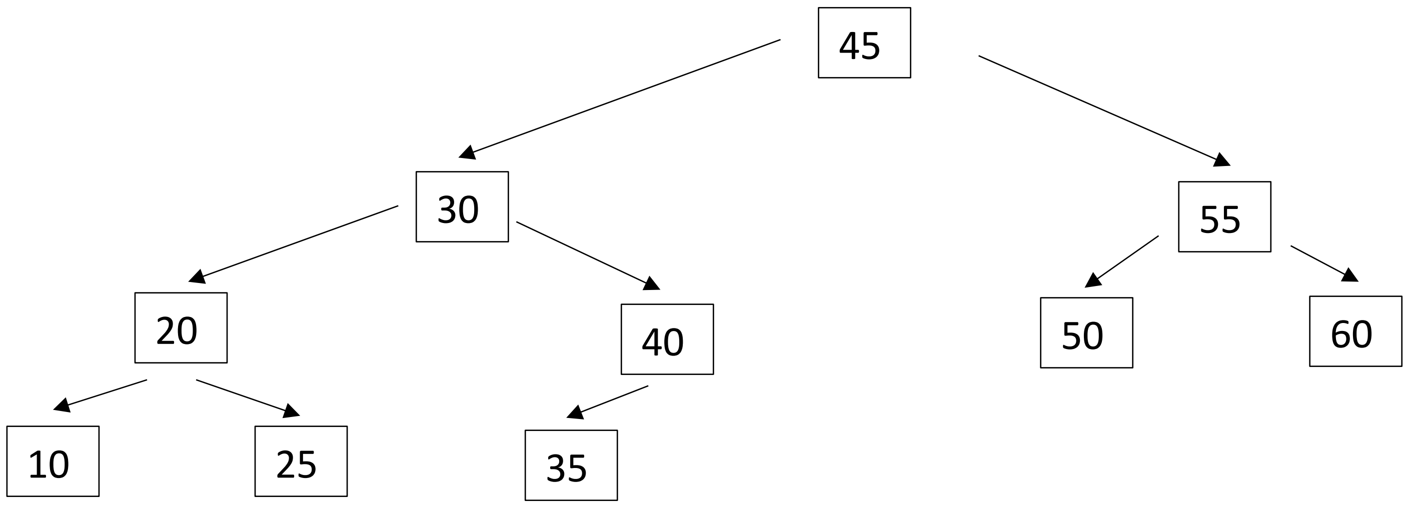
a-) 5,7,6,1



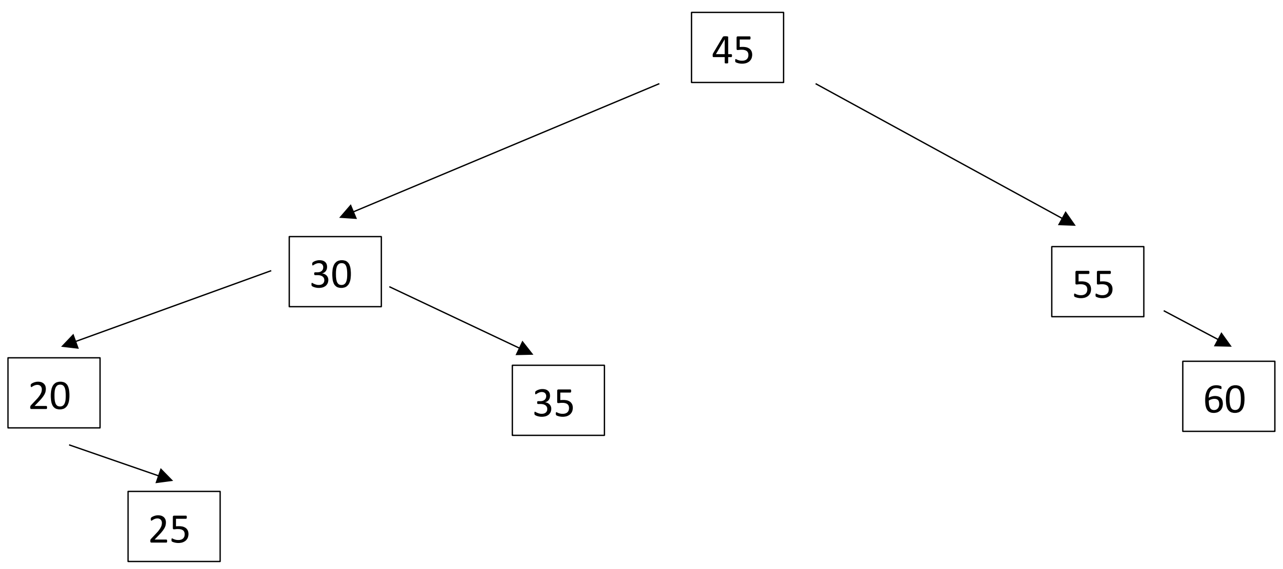




b-) 40, 50, 45, 30, 60, 55, 20, 35, 10, 25



Delete 10, 40, 50



Question-2

Data Structure	insert	extractMin
unsorted array	O (1)	O(n)
AVL tree	O($\log_2$n)	O($\log_2$n)
min-heap	O($\log_2$n)	O($\log_2$n)
unsorted linked list	O (1)	O(n)
sorted linked list	O(n)	O(n) or O(1) !!!

!!! There is two different run time because if linked list sorted in ascending order then min value bill be first node and removing first node is O(1).

Question-4

1-) Both getLessThan and getGreatherThan have same running time because they make almost same thing. Their running times are O (1) in best case where root is equal to search key. Their worst case is O(n) because we have to check all the numbers in the heap and we use traversal(preorder).

When we look to their average cases, their running times are between O (1) and O(n) because we know that for min-heap (getLessThan method), parent always smaller than its children and if parent larger than search key we do not have to check its sub-heaps (the situation is reverse for max-heap and getGreatherThan method).

We cannot do this asymptotically better because already program checks min number of items.

2-) To get the median I use both min-heap and max-heap. Min-heap to keep items greater than current median and max-heap to keep items less than current median. Current median is median among items inserted that time. The purpose of that implementation is keeping median in the root of the heaps or heap. When the different between max and min heap larger than 1, method removes the root of larger heap and insert it to smaller heap. The reason of why we transferring the root is to keep balance and keep the roots in the middle of the numbers (linear).

The running time of insertion is “3.log(n) + 3” = O (log n)

After insertions done, getting median is simple. Get the larger heap’s root, and if their sizes equal than median is average of the heaps’ roots.

The running time of getting roots is “3” = O (1)