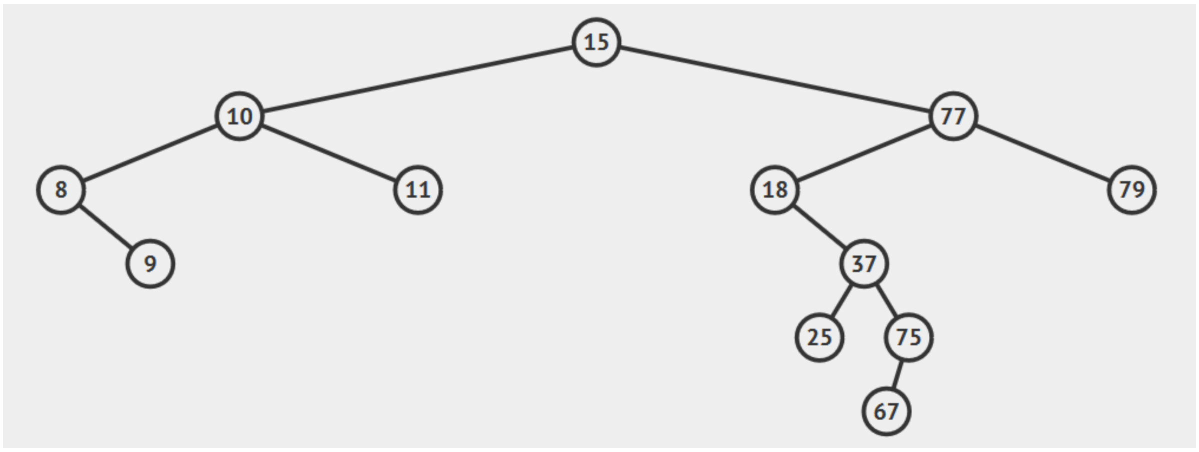
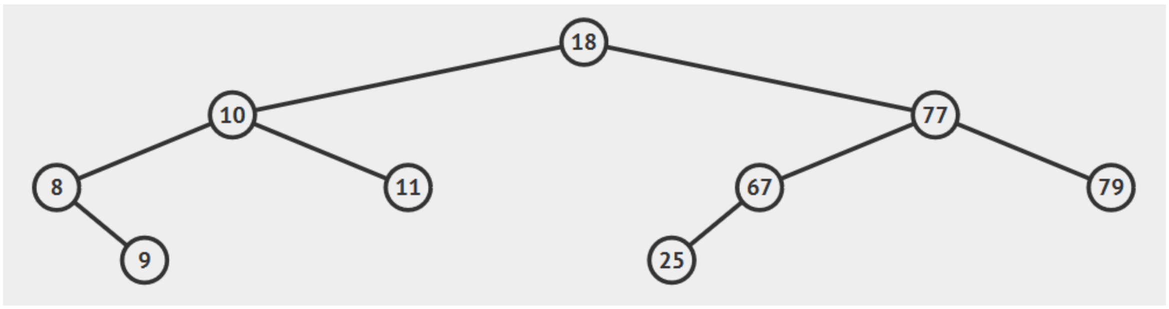
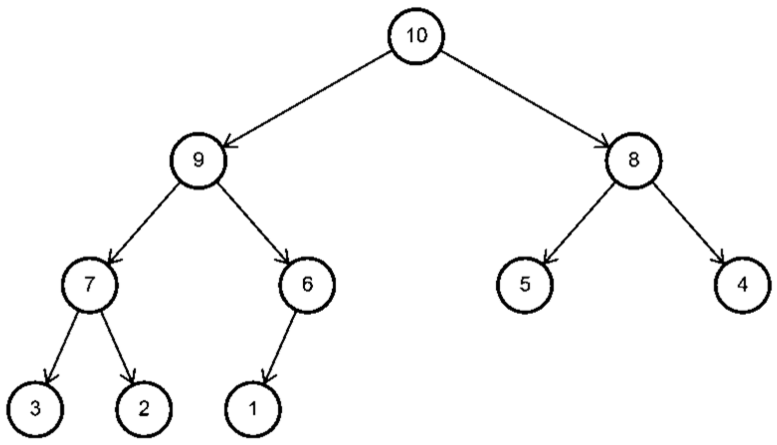
Part-1a) The AVL tree with values 15, 77, 79, 10, 18, 8, 11, 37, 25, 75, 9, 67 inserted into an empty one in given order looks like the following:



After deleting 37, 15, 75 in the given order from this AVL tree we get the following tree:



Part-1b) The min-heap after inserting 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1, 3 in given order will look like the following:



Part-1c) (1) All binary search trees are heaps. – This statement is false, because the heap property does not cover the BST property. The rule for a binary search tree is that all the elements in the left subtree of the root is smaller than the root, whereas all the elements in its right subtree is greater than the root. On the contrary, a heap requires that the root be smaller/greater (minheap/maxheap) than all its descendants, and this rule recursively applies to all the nodes in that heap. A binary search tree root will have values as its children both smaller (on the left subtree) and greater (on the right subtree), and this property will recursively apply to all nodes in the BST; thus, neither being a minheap nor being a maxheap.

(2) The depths of any two leaves in a maxheap differ by at most 1. – This statement is true. A heap is designed to insert new elements into it such that it always protects the property of being a complete binary tree. When a new node is added into the binary heap, it is added to the leftmost empty slot available in the last level, this way it keeps the property of being a complete binary tree. Similarly, after each delete operation the heap rebuilds itself to hold its complete binary tree property. Since the depths of any two leaves in a complete binary tree differ by at most 1 and since all heaps are complete binary trees, this statement holds.

(3) In an AVL tree, conducting a left rotation on a node and then a right rotation on the same node will not alter the overall tree structure. – This statement is false. The double-rotations are made to preserve the property of an AVL only when it loses it after an insertion/deletion. When the AVL tree readily possesses the AVL properties (namely for and node in the tree, the height of the left and right subtrees differ by at most 1) doing a double rotation will corrupt this property by creating a height difference in the left and right subtrees greater than 1 on the node which the double rotation was applied.