1. (5 points) An AVL tree is a binary search tree that is nearly height balanced, meaning that for each node \( x \) in the tree, the heights of the left and right subtrees of \( x \) differ by at most 1. One way of implementing an AVL tree, is to maintain 2 extra attributes in each node: \( x.left.h \) is set to the height of the left subtree of \( x \) and \( x.right.h \) is set to the height of the right subtree of \( x \).

To insert a key into an AVL tree, first place a node into the appropriate place with binary search tree insert. Afterward, the tree might no longer be height balanced. Specifically, the heights of the left and right children of some nodes might differ by more than 1.

Insert the keys \( \langle 62, 18, 46, 42, 39, 3, 10 \rangle \) into an AVL tree. Show \( x.left.h \) and \( x.right.h \) as \( (x.left.h, x.right.h) \) at each node after a key is inserted. If the key insertion causes a right or left rotation, show the intermediate steps and left and right subtree heights at each node after each rotation. For example, if a left rotation is followed by a right rotation, show both intermediate trees. Note: You can show the left and right height in a set of parentheses, where \( (1, 0) \) means the left subtree is height one and the right subtree is height 0.
2. (5 points) A 2-3 tree is a height-balanced binary search tree, in which every node can hold either 1 or 2 keys. The first type, containing only 1 key value, is called a two-node and is a normal bst node with 1 key and 2 children. The other type, containing 2 key values, is called a three-node, and can have 3 children, one less than the left key in the node, one greater than the right key in the node, and one whose value lies between the left and right key values in the node. Nodes are inserted as they are in a normal bst, except that the inserted value may end up as the third key in a node. When a node contains 3 values, it is overloaded, in which case the middle value is pushed or promoted up to the parent node. If the parent node is then overloaded, the middle value is pushed higher in the tree. When a node is promoted, what used to be its left and right values in the three-node become two-nodes.

Insert the keys (62, 18, 46, 42, 39, 33, 90, 75, 8) in order from left to right into an initially empty 2-3 tree.