README

BST Class and Constructor
To start off, I created a bst (binary search tree) class, and I also made a constructor which creates an empty bst by setting the root node to null.

Insert
This method inserts nodes into the bst. The method does this by first figuring out where to put the new node by comparing the new node to the current node in the tree. The method then traverses the bst to the left if the new node is less than the current node or it traverses the bst to the right if the new node is greater than the current node. If it is found that the parent node is null after traversing the bst, that means the bst is empty so the method sets the root node to the new node. If the bst is not empty then sets the parent's left child to the new node if the new node is less than the parent node, or it sets the parent's right child to the new node if the new node is greater than the parent node.

In order, Preorder, and Postorder
These methods traverse the bst in three ways; in order, preorder, and postorder. The in order method traverses the bst by going to the left node first, then the root node, then the right node and it keeps this pattern until the entire bst has been traversed. The preorder method traverses the bst by going root, left, then right, and the postorder method traverses the bst by going left, right, then root.

Search
This method searches the bst for a specified node. Just like inserting a node, if the specified node is less than the current node then the method traverses the bst to the left, or it traverses to the right if the specified node is greater than the current node. The method traverses the bst until the specified node is found or until it reaches the end of the bst and does not find the node in which case it returns null.

Height
This method finds the height of the bst. First the method checks if the root node is null and if that is the case then the bst is empty and it returns zero. If the bst is not empty then the method traverses the bst all the way to the left and keeps track of the number of levels, and it also traverses the tree all the way to the right and keeps track of the levels. The method then compares the number of levels on the left to the number of levels on the right, and whichever is greater, the method returns that value plus one to account for the root node.

Minimum and Maximum
These two methods find the minimum and maximum values in the bst. The minimum method traverses the bst all the way to the left and returns that last value because the smallest value in a bst will be all the way to the left. The maximum method traverses the bst all the way to the right and returns the last value which will be the largest value in the bst.

Nth Largest (with helper method)
These methods are used to find the Nth largest node in the bst. The helper method takes the contents in the bst and stores them in an arraylist in ascending order. The "Nth Largerest" method first checks if the bst is empty and if it is then it returns -1, otherwise we use the helper method to store the nodes in an arraylist using the helper method. By using a loop, the method starts at the bottom of the arraylist and increments by one until it finds the Nth largest node. (N is provided by the user).

Delete
This method is for deleting a node. Just like in a couple of other methods, this method traverses the bst to the left if the target node is less than the current node, or traverses to the right if the target node is greater than the current node. If the target node is found, there will be three cases. The first case is that the target node has no children. In this case, the method deletes the target node by setting it to null. In this first case it is also possible that the target node is the root node, if so, the method sets the root node to null. The second case is that the target node has two children, which means some tree reorganization will be required. In this case, the method finds a "successor" node that has the smallest value of all the nodes that are greater than the target node. The "successor" node is then deleted and the value of the target node is set to the value of the "successor" node. The third case is that the target node has one child. In this case, the method deletes the target node by setting the child of the parent's target node to the target node's child.

BFS and DFS
These two methods do a breadth first search and a depth first search of the bst. The bfs method prints out each level of the bst from the top (by using a helper method) and then traverses each level from left to right. The output of the dfs method is the same as the preorder method so the "dfs" method is actually just the same call as the preorder method.