ADS HW 8 BONUS

Problem 8.2

C) (Bonus)

The technique implemented in the code constructs the tree from leaves to root. At first, the number of nodes in the list is counted, and then half the nodes are taken to construct the left subtree. Then, the middle node is assigned as the root and the remaining half of the nodes are used to construct the right subtree (since the linked list is already sorted, the BST should be correct when implemented this way).

To find an element that is at the end of the tree, we must traverse all elements from the root to the leaf level. So, searching in binary search tree has time complexity O(h) where 'h' is the height of the BST. However, a linked list can only be traversed from node to node until the node we seek is reached making time complexity for searching O(n). As such, searching for a specific value in a linked list requires traversing all the elements until we find that element making search time complexity O(n). The reason for this implementation is that we want the search time complexity for the BST to be smaller than for the linked list and we achieve that because of the height of the binary tree being half the number of elements(nodes) of the linked list making for smaller time complexity in comparison meaning since h = n/2, O(h) = O(n/2) of the BST has smaller search time complexity than the linked list's O(n) where 'n' is the total number of nodes. (root needs to be the middle node)

References:
https://www.geeksforgeeks.org/sorted-linked-list-to-balanced-bst/