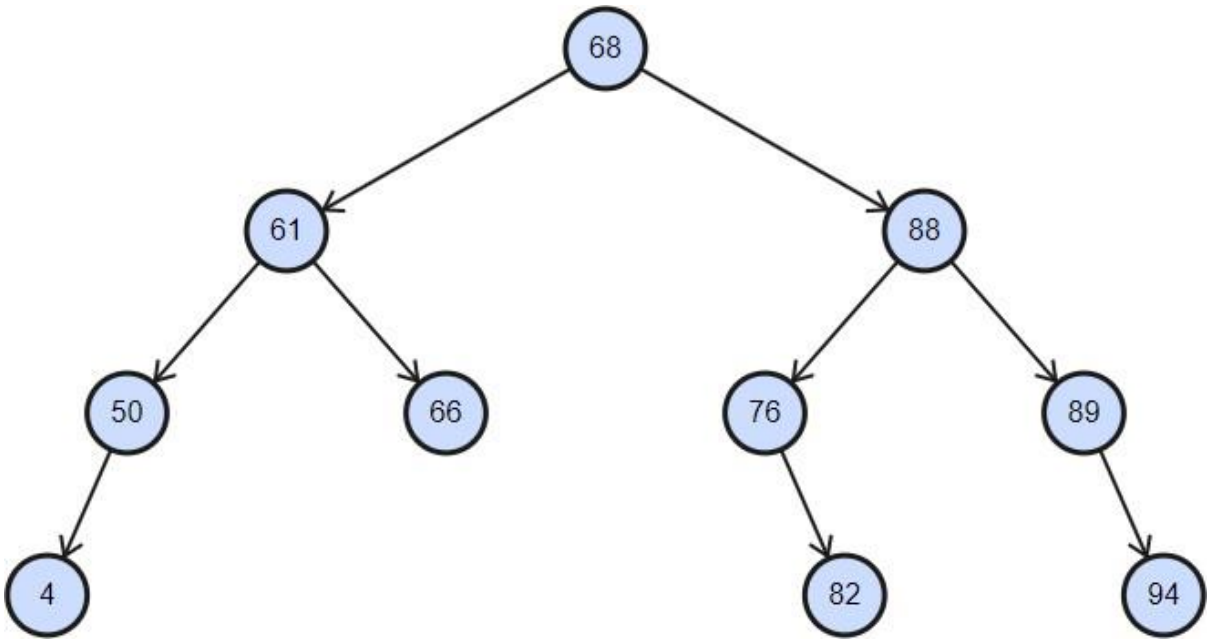
Question 1:

a) Using the helper function insert (bst, key), create the binary search tree that results from inserting the following keys in the order given: 68, 88, 61, 89, 94, 50, 4, 76, 66, and 82.

{'value': 68, 'left': {'value': 61, 'left': {'value': 50, 'left': {'value': 4, 'left': {}, 'right': {}}, 'right': {}}, 'right': {'value': 66, 'left': {}, 'right': {}}}, 'right': {'value': 88, 'left': {'value': 76, 'left': {}, 'right': {'value': 82, 'left': {}, 'right': {}}}, 'right': {'value': 89, 'left': {}, 'right': {'value': 94, 'left': {}, 'right': {}}}}}



b) Using the helper function exist (bst, key), check whether key 50 exists in resultant Binary Search Tree.

True

c) Using the helper function exist (bst, key), check whether key 49 exists in resultant Binary Search Tree.

False

d) Using the helper function minimum (bst, starting_node), find the node with the minimum value in resultant Binary Search Tree from starting node = 68.

4

e) Using the helper function minimum (bst, starting_node), find the node with the minimum value in resultant Binary Search Tree from starting node = 88.

76

f) Using the helper function maximum (bst, starting_node), find the node with the maximum value in resultant Binary Search Tree from starting node = 68.

94

g) Using the helper function maximum (bst, starting_node), find the node with the maximum value in resultant Binary Search Tree from starting node = 61.

66

h) Using the helper function inorder_traversal (bst), perform in-order traversal of the Binary Search Tree.

4, 50, 61, 66, 68, 76, 82, 88, 89, 94

i) Using the helper function preorder_traversal (bst), perform pre-order traversal of the Binary Search Tree.

68, 61, 50, 4, 66, 88, 76, 82, 89, 94

j) Using the helper function postorder_traversal (bst), perform post-order traversal of the Binary Search Tree.

4, 50, 66, 61, 82, 76, 94, 89, 88, 68