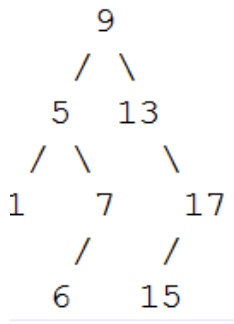
You are given the following tree:



1. Write the function which traverses the whole tree and has the output: *1 5 6 7 9 13 15 17* – ***1p***
2. Write the function which calculates the maximum depth of this tree. – ***1p***
3. Write the function which searches for a node in the tree and outputs 1 if it found it and 0 if it did not. – ***2p***
4. Write the function which returns 1 if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum. Return false if no such path can be found.–***2p***
5. For each node in a binary search tree, create a new duplicate node, and insert the duplicate as the left child of the original node. The resulting tree should still be a binary search tree. – ***2p***
6. What is the complexity of the following operations on P.B.B.S.Ts: *search, insert*. – ***1p***

Example of input/output:

Function1(root)	1 5 6 7 9 13 15 17
int maxDepth = maxDepth(root);	Max Depth = 4

int found;
found = search(root, 5);
found ? printf("yes!") : printf("no!");
found = search(root, 10);
found ? printf("yes!") : printf("no!");
	yes!
	no!

int hasPathSum
int pathSum = 27;
hasPath = hasPathSum(root, pathSum);
hasPath ? printf("%d is pathSum",pathSum) : printf("%d is not pathSum",pathSum);
pathSum = 16;
hasPath = hasPathSum(root, pathSum);
hasPath ? printf("%d is pathSum",pathSum) : printf("%d is not
	27 is path sum
	16 is not path sum

doubleTree(root); Function1(root);	1 1 5 5 6 6 7 7 9 9 13 13 15 15 17 17