BINARY TREES

Create two separate classes, one for a simple binary tree and one for a binary search tree. Implement questions 2, 3, 5 and 6 in the binary tree class and questions 1 and 4 in the binary search tree class.

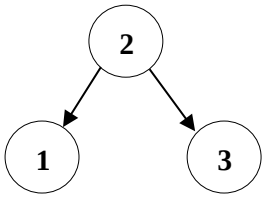
Question 1: Given a sorted (increasing order) array with unique integer elements, write a function to create a binary search tree with minimal height. The array will be passed as an argument to the function and the function would return the height of the created tree.

Question 2: Given a binary tree, write a function which creates a linked list of all the nodes at each depth (e.g., if you have a tree with depth D, you'll have D linked lists). The tree will be passed as an argument to the function and the function will return the number of linked lists created (depth of the tree).

Question 3: Implement a function to check if a binary tree is balanced. A balanced tree is defined to be a tree such that the heights of the two subtrees of any node never differ by more than one. The tree will be passed as an argument to the function and the function will return true if the tree is balanced, otherwise it will return false.

Question 4: A binary search tree was created by traversing through an array of elements from left to right and inserting each element. Given a binary search tree with distinct elements, print all possible arrays that could have led to this tree. The tree is passed as an argument to the function and the possible arrays are printed.

EXAMPLE Input:



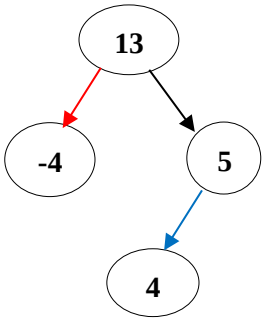
Output: {2, 1, 3}, {2, 3, 1}

Question 5: Tl and T2 are two binary trees, with Tl bigger than T2. Create a function to determine if T2 is a subtree of Tl. A tree T2 is a subtree of Tl if there exists a node n in Tl such that the subtree of n is identical to T2. That is, if you cut off the tree at node n, the two trees would be identical. Both trees are passed as an argument to the function. The function will return true is T2 is a subtree of T1, otherwise it will return false.

Question 6: You are given a binary tree in which each node contains an integer value (which might be positive or negative). Write a function to count the number of paths that sum to a given value. The path does not need to start or end at the root or a leaf, but it must go downwards (traveling only from parent nodes to child nodes). The tree and value are passed as an argument to the function. The path count is returned by the function.
Sum of a path is the sum of all node values in that path.

EXAMPLE Input:

Tree =
Value = 9



Output: 2 path