Muhammad Ismail
20I-0941

TASK NO 1

This function is implemented in the way that the mid element of the array is taken as the root as the array is sorted which is described in the problem statement. Then we Loop through the array and insert the element one by one in the tree according to the rules of the BST. As the result of this we get the tree which has minimum height.

In order to calculate the Height, we use the recursion. It's like we say the left subtree go and give me your height then we ask the right subtree to go and give me your height then after getting the height of both left and right subtree we find whose height is maximum and then add one to it to get the final height.

TASK NO 2

In Order to generate the linked list, I used the Queue data structure. In simple words, I push the parent element in the Queue then pop it and then add the popped element to the Linked list. Then the iteration goes on until the children of each node are added into the linked list for particular breath.

It's like Get the root element and push it into the queue then generate the linked list out of it then again do the same but this time for the children of the root element as they will be pushed into the queue during the iteration. The Linked list pointer is set by using the previous element and set's it's next pointer to the child of the Node (breath wise). After he creation of each Node they are displayed using the display function of the linked list.

TASK NO 3

In this task, using recursion I get the height of the left subtree and then the right subtree. By subtree I mean each node as we are doing it recursively. Then after getting the height of each node, we calculated the difference between them if the difference between the height was {-1,0,1} then we set the balance factor as true and did not return false and goes ahead. If any of the node does not satisfy the above property then the function would return false which means they are not balanced.

TASK NO 4

Tried my best but was unable to do it! Changed code many many times but in vain. However, realized the importance of math's for programming. Dummy tries are included but they do not do much.

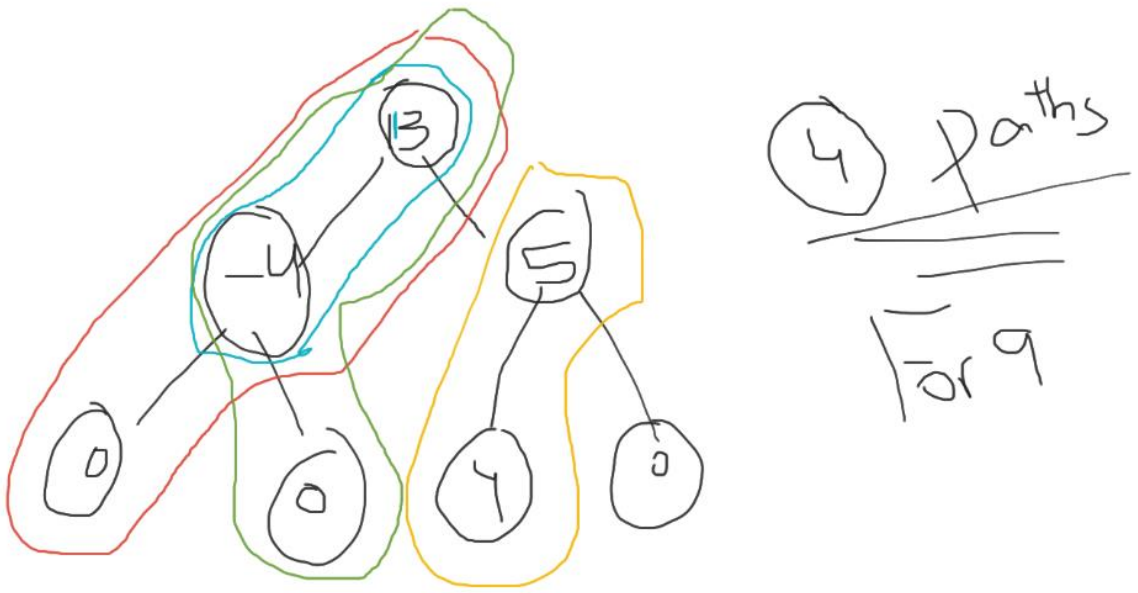
My idea was to traverse the BST in every possible way but that did not work. The next idea was to create an array from the BST then apply operations on that particular array. Got it correct in my calculation but left somewhere while coding. Hopefully next time!

TASK NO 5

In this task, T1 is bigger and T2 needs to be get checked. So, at first, we check if T2 is null if it is then it is subtree. Similarly, if T1 is null and T2 is not then it is not subtree. Then we call a function that checks if both the tree is same. If they are good, we return true. Otherwise, we move one step next to the T1 and then again repeat the above steps and process goes on until they are matched or in the end, we return false indication tree do not match

TASK NO 6

Checking the path sum was like using 2 pointers. One would stay on the root while the other go and counts the sum of the tree if it found any match then it increases the count. Similarly, The first pointer goes on increasing one by one while the second pointer in each iteration of the first pointer goes and check the complete tree weather is it path or not? If it is path then it increment it otherwise it do not increase.


13
-4
5
0
0
4
0
4 paths
for 9