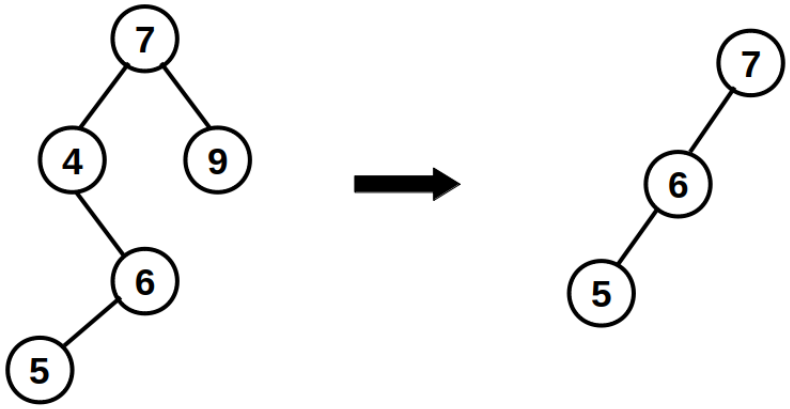
Algorithms Laboratory (CS29203)
Assignment 5: Tree, Binary search tree
Department of CSE, IIT Kharagpur

12th October 2023

Question-1 (50 points)

In this question, we will be modifying a binary search tree (BST) by removing some nodes and maintaining its structure. Suppose that you are given a BST having integer values ≥ 0 in its nodes. Now we want to keep the nodes int the BST which lie in the range [min, max]. After deleting the nodes beyond this rage, the relative structure of other nodes should remain the same. That is, any node's descendant should remain a descendant after the deletion operation. Note that after deletion of specific nodes, the resultant BST will be unique.

For example, consider the BST in the left side of the following figure. Let the range of values to keep are $[5, 7]$. Then the resultant BST will be the one shown in the right side of the figure.



You can assume that the input is taken in an array in the following format: [7, 4, 9, -1, 6, -1, -1, 5], where the nodes are defined in level order from left to right and a missing child is denoted as '-1'. Your task is to develop and implement an algorithm to solve the above problem. You can assume that each node in the tree is unique.

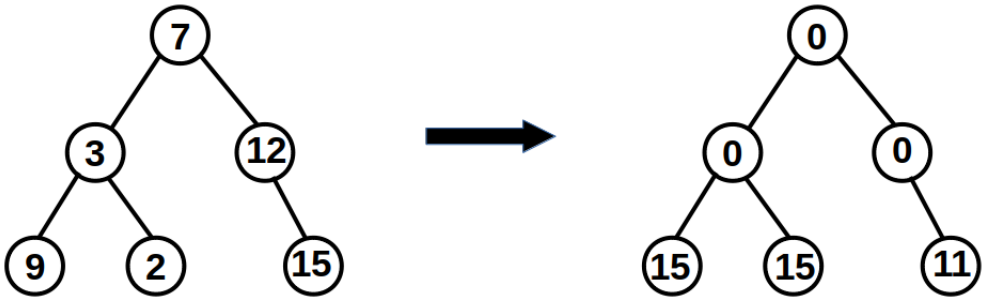
Important: You are **not** allowed to implement array based representation of trees (you have to use standard linked representation of trees). To get the full credit, you are **not** allowed to create a new tree, you have to modify the existing tree accordingly.

Example:

```
(Input)  [7, 4, 9, -1, 6, -1, -1, 5], min = 5, max = 7

(Output) [7, 6, -1, 5]



(Input) [5, 3, 7], min = 5, max = 7

(Output) [5, -1, 7]
```

Question-2 (50 points)

Let us consider a rooted binary tree where the nodes can have integer values ≥ 0. We will be performing some modifications of the values of the nodes in the tree in the following way. First we define two nodes of the tree as *cousins* if they are in the same level, but have different parents. Our goal is to change the value of each node in the tree to the sum of the values of it's cousin nodes. For example consider the left part of the following figure. In the tree, the root node does not have any cousin, so in the modified tree it's value will be 0. The next level nodes 3 and 12 also does not have any cousin, hence the corresponding values will also be 0. Now the node with value 9 has one cousin with value 15, hence this node's value will be modified to 15. Similarly the node with value 2 will be modified to 15 for the same reason. Finally the node with value 15 has two cousins: nodes with values 9 and 2. Hence this node will be modified to $9 + 2 = 11$. This is what is shown in the right part of the figure.



You can assume that the input is taken in an array in the following format: [7, 3, 12, 9, 2, -1, 15], where the nodes are defined in level order from left to right and a missing child is denoted as '-1'. Your task is to develop and implement an algorithm to solve the above problem.

You are **not** allowed to implement array based representation of trees. To get full credit, your implementation should have worst case complexity of $O(n)$ where n is the number of nodes in the tree.

Important: You are **not** allowed to implement array based representation of trees (you have to use standard linked representation of trees). To get the full credit, you are **not** allowed to create a new tree, you have to modify the existing tree accordingly.

Example:

```
(Input)  [2, 7, 11, 6, -1, 5, -1, -1, 19, -1, 3, 1, 10, 4, 20]

(Output) [0, 0, 0, 5, -1, 6, -1, -1, 3, -1, 19, 24, 24, 11, 11]



(Input) [12, 5, 10]

(Output) [0, 0, 0]
```