# Assignment 3

You must solve all the problems below, preferably before your lab test. The lab evaluation questions may build upon these assignment questions, and therefore it is strongly recommended that you try these questions beforehand.

Some test cases are provided here and in the skeleton code supplied to you. Feel free to add your own test cases to the skeleton code for testing purposes.

## Question 1

Write a function to insert a new value into the given AVL tree. Make sure that the tree stays balanced.

There are four methods that you have to implement:

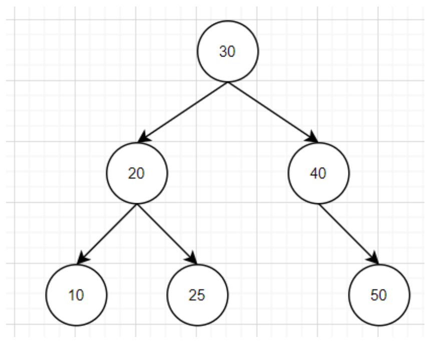
1. Node rightRotate(Node y): This performs a right rotation at the given node and returns the updated root of rotation.
2. Node leftRotate(Node x): This performs a left rotation at the given node and returns the updated root of rotation.
3. int getBalance(Node N): This returns the balance factor (difference in heights of left and right subtrees respectively) of the given node.
4. Node insert(Node node, int key): This returns the root of the AVL tree after performing the insertion of key and balancing.

Feel free to write any helper functions that you may require such as a function to calculate heights of subtrees.

### Sample Test Cases:

Test 1:

Sequence of insertions: 10 20 30 40 50 25
Created Tree:



## Question 2

You had an MST, but your friend, in an effort to annoy you, moved around some nodes in the tree such that the keys don't really hold significance anymore as nodes have been jumbled across the tree. However, there would still be subtrees that are valid MSTs. You want to see what is the greatest total of any valid MST that you still have left.

Given a multiway tree root, return the maximum sum of all keys of any subtree which is also a multiway search tree. You can assume that the key values are always positive.

There is only one method to implement:

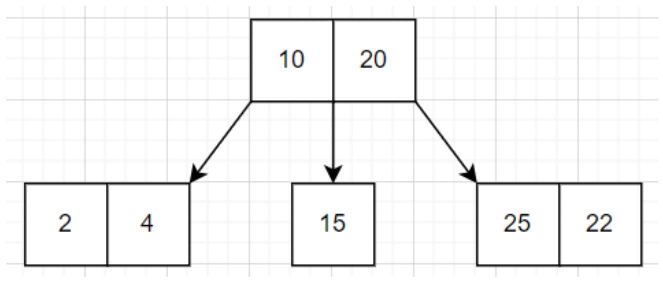
1. public static int getMaxMST(Node temp): This takes as input the root node of the multiway tree and returns the max sum of all keys of a subtree which is a valid MST.

Feel free to write any helper functions that you may require such as a function to calculate heights of subtrees.

### Sample Test Cases:

Test 1:

Input MST:

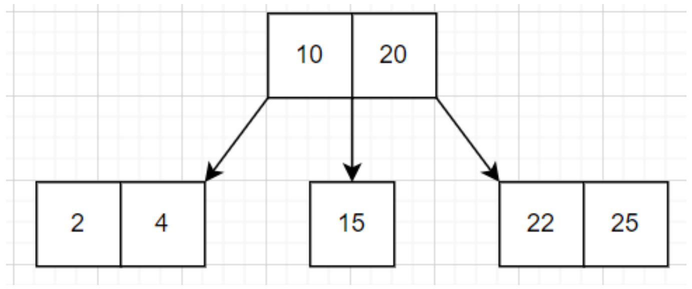


Output: 15

Explanation: Since the node (25,22) forms an invalid MST, the subtree which is a valid MST and has the greatest possible sum of keys is 15.

Test 2:

Input MST:



Output: 98
Explanation: This entire tree is a valid MST. Hence, the max sum of keys is 98.

## Question 3

You will be given two arrays, H and W. H[i] and W[i] represent the height and weight of the ith person. Your task is to sort both lists such that the people are in ascending order of height. If multiple people have the same height, those people should be sorted in increasing order of weight. Your solution must be O(n log n).

### Sample Test Case

H = [3,4,1,7,8,4]
W = [2,5,6,3,7,1]

After sorting:
H = [1,3,4,4,7,8]
W = [6,2,1,5,3,7]

Explanation:
Heights are sorted in increasing order, and weights are sorted correspondingly (weight of person with height 1 is 6, so 6 is first element in W' array).
For the two people with height 4, weights should be in ascending order, so we have 1 followed by 5.

This question has been set up on Moodle for you to practice.

## Question 4

Given an array A and a target array, your task is to make A **similar** to the target array.
Two arrays are considered similar if the frequency of each element is the same. For example, the arrays [1,1,1,3,3,4] and [3,4,1,1,3,1] are similar. However, the arrays [1,2,3,4] and [1,1,2,3] are not similar - the frequency of 4 in the first array is 1, while in the second array it is 0.

You are only allowed to perform certain operations on the array. As part of one operation, you will:

- select two indices (i,j) (no constraints on *i* and *j*).
- Increment the element at the ith position by 2: A[i] = A[i]+2
- Decrement the element at the jth position by 2: A[j] = A[j]-2

You can only perform operations on the array A. You cannot perform partial operations - that is, you have to increment the element at *i*, and simultaneously decrement the element at *j*, you cannot do one of the two.

If it is not possible to make the array A similar to the target array, return -1; otherwise return the minimum number of operations to make A similar to the target.

### Sample Test:

A = [1,2,5]; Target = [4,1,3]
Expected output: 1
Explanation: select i = 1, j = 2 (0- indexed). After incrementing A[1] by 2, and decrementing A[2] by 2, we have A = [1,4,3], which is similar to the target array.

A separate driverCode has been provided for this question. The other questions have been tested in the main function of the same file where you implement the required methods.

## Question 5

You are given the root of a binary tree. Invert the tree horizontally, turning it from left to right, such that the left child becomes the right child and the right child becomes the left child for all nodes.

Input:

- The input root is a reference to the root of a binary tree.

Output:

- Return the reference to the root of the inverted binary tree.

Sample Test:

```
    1
   / \
  2   3
 /\  /\
4 5  6 7
```

After inverting:

```
    1
   / \
  3   2
 /\  /\
 7 6 5 4.
```

This question has been set up on Moodle for you to practice.

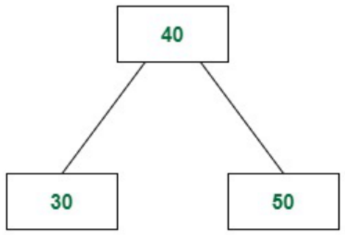
## Question 6

Given a 2-4 tree, and a range [L, R], remove all keys less than L, and all keys greater than R, and return the resulting 2-4 tree. Your algorithm must be O(n).

Sample Test:

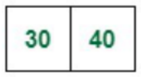
Test 1:

Input tree:



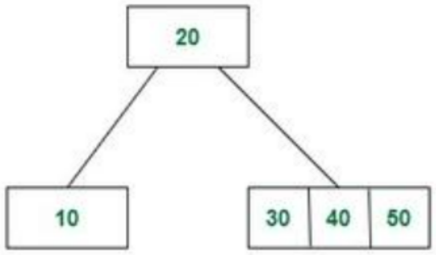
[L, R] = [25,45]

Output tree:



Test 2:

Input tree:



[L,R] = [15,55]

Output tree: