Assignment V
Lab MC504

Send your assignment solution to mc504lab@gmail.com.
Deadline: 10.02.2021, 12 midnight.
Put all files into one folder, create a zip and name it as <**RollNo>_<Assignment_<No>** and mention the files name as: Q1.c, Q2.c and so on. In each file please mention your roll number.
Subject of mail should be: <**RollNo>_Assignment_<No>**. **For example : 1911MC04_Assignment_II.**
You have to take inputs from the user. Otherwise marks (40%) will be deducted.

1Q.

Write a C program to implement a Binary Search Tree and print a tree in level order traversal.
Suppose there are total eight nodes
12, 5, 7, 1, 10, 8, 20, 3 **(It is just for example ; you have to take i/p from the user.)**

Make 10 as root and develop a BS tree. **(It is just for example ; you have to take i/p from the user.)**

Print level order traversal.
Now delete a node and print level order traversal.
Now ask the user to delete a node. If possible to delete, then perform delete operation otherwise print “not possible to delete.”
Ask from user to search for any key, if found the key print found successfully !! otherwise print “not found.”

2Q.

Write a C program to check whether a square matrix is a magic square or not. Take the dimension and the members of the matrix from the user.

A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2.

Example :
Magic Square of size 3
---
2 7 6

9 5 1
4 3 8
Sum in each row & each column = 3*(3^2+1)/2 = 15

3Q
An AVL tree (Georgy Adelson-Velsky and Landis' tree, named after the inventors) is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

We define balance factor for each node as :

balanceFactor = height(left subtree) - height(right subtree)

The balance factor of any node of an AVL tree is in the integer range [-1,+1]. If after any modification in the tree, the balance factor becomes less than −1 or greater than +1, the subtree rooted at this node is unbalanced, and a rotation is needed.

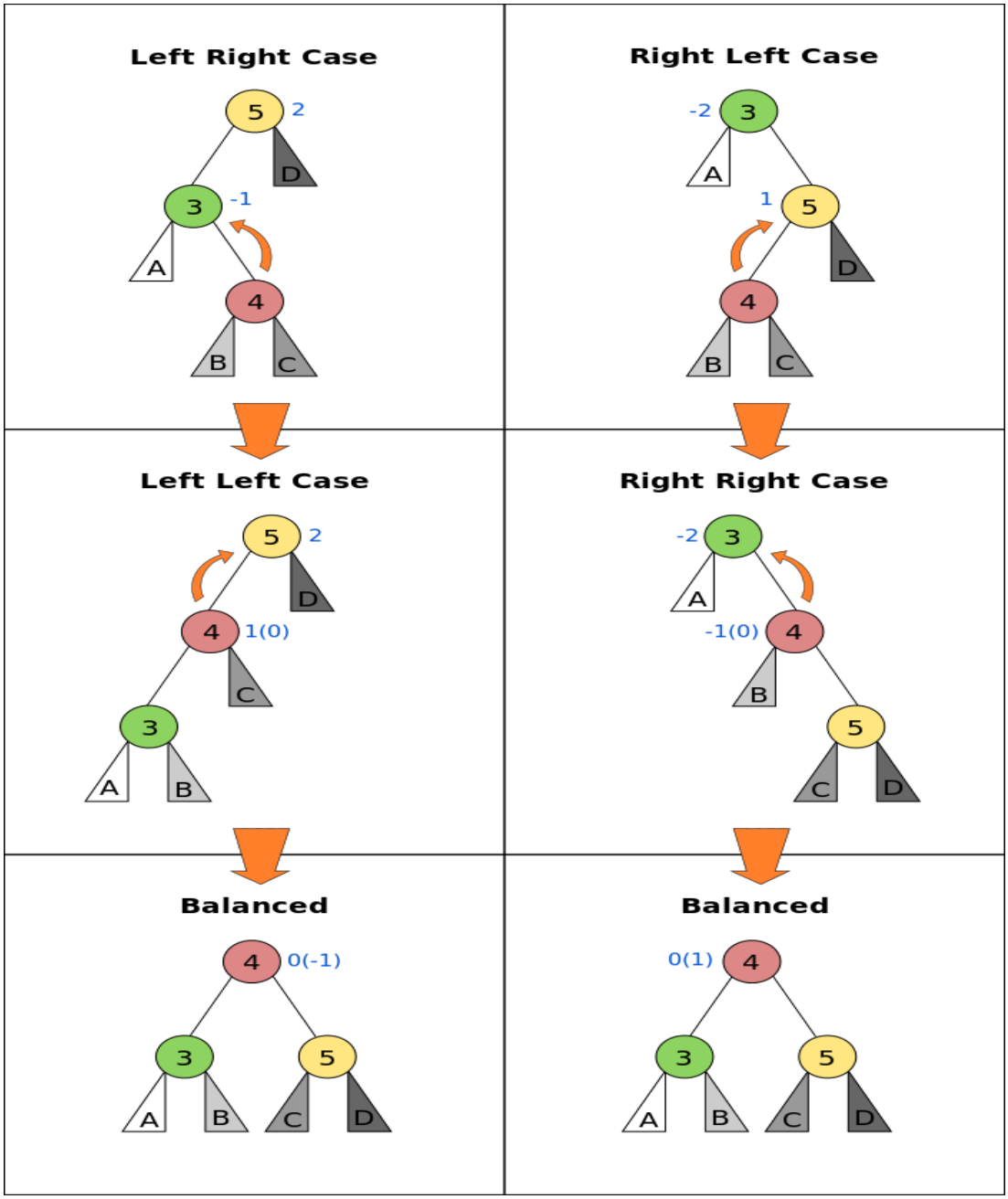
(https://en.wikipedia.org/wiki/AVL_tree)

You are given a pointer to the root of an AVL tree. You need to insert a value into this tree and perform the necessary rotations to ensure that it remains balanced.

Node is defined as

```
struct node
{
int val;          //value
struct node* left;  //left child
struct node* right; //right child
int ht;           //height of the node
} node;
```

Note: All the values in the tree will be distinct. Height of a Null node is -1 and the height of the leaf node is 0.



Output :

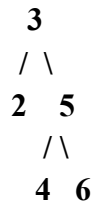
Insert the new value into the tree and return a pointer to the root of the tree. Ensure that the tree remains balanced.

Sample INPUT

```
  3
 / \
2   4
     \
      5
```

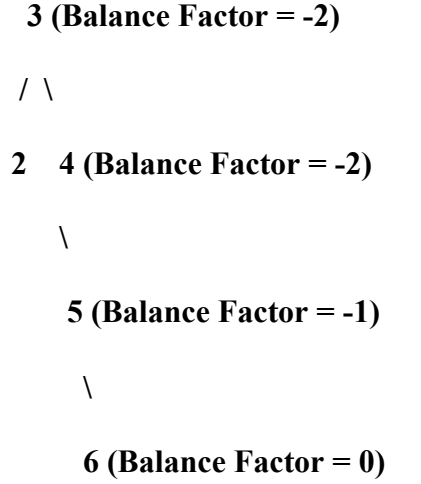
The value to be inserted is 6.

Sample Output



Explanation

After inserting 6 in the tree. the tree becomes



Balance Factor of nodes 3 and 4 is no longer in the range [-1,1]. We need to perform a rotation to balance the tree. This is the right case. We perform a single rotation to balance the tree.

After performing the rotation, the tree becomes :