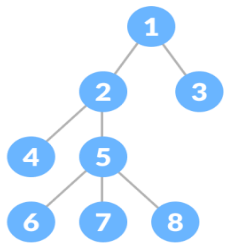
Q.1) Define following terms

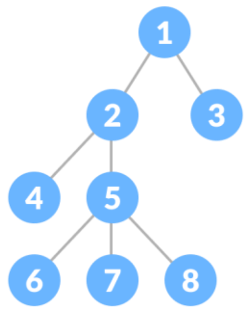
1) Tree:

- ✓ Tree is a hierarchical data structure which stores the information naturally in the form of hierarchy style.
- ✓ It is a non-linear data structure compared to arrays, linked lists, stack and queue.
- ✓ It represents the nodes connected by edges.



2) Root Node:

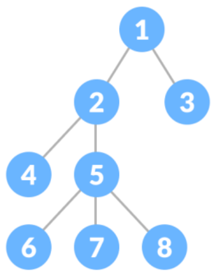
- ✓ The root node is the topmost node in the tree hierarchy.
- ✓ In other words, the root node is the one which doesn't have any parent.



- ✓ In Above Example 1 is root node.

3) Leaf Node

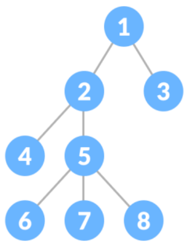
- ✓ The node which does not have any child node is called the leaf node.
- ✓ Leaf node is the bottom most node of the tree.



- ✓ In Above Example 6,7 ,8 all are Leaf Node

4) Path:

- ✓ Path refers to the sequence of nodes along the edges of a tree.

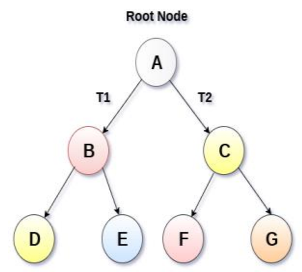


- ✓ In Above Example 1-2-4 is a path

  1-2-5-7 is a path

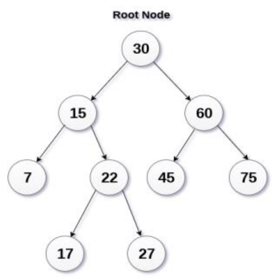
5) Binary tree:

- ✓ A binary tree is a special type of tree in which every node or vertex has either no child node or one child node or two child nodes.



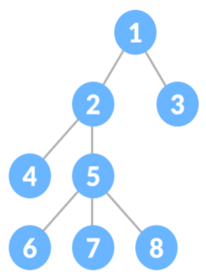
6) Binary Search tree:

- ✓ Binary Search tree in which the nodes are arranged in a specific order.
- ✓ This is also called ordered binary tree.
- ✓ In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.
- ✓ Similarly, value of all the nodes in the right sub-tree is greater than or equal to the value of the root.



7) Degree of a node:

- ✓ In a tree data structure the total number of child of a node is called as degree

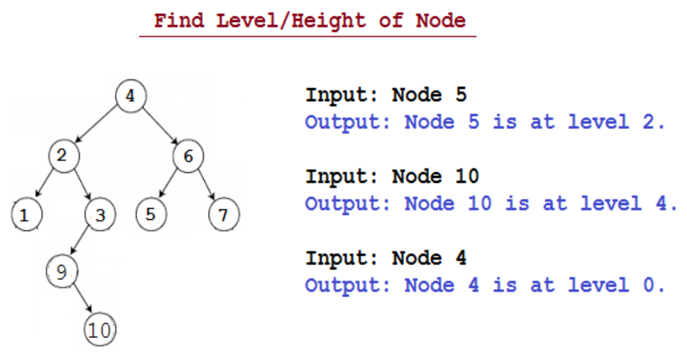


Degree of 1 is 2

Degree of 3 is o
Degree of 5 is 3

8) Level of a node:

✓ In a tree data structure the root node is said to be level 0 and the children of root node are at Level 1 and so.



Find Level/Height of Node

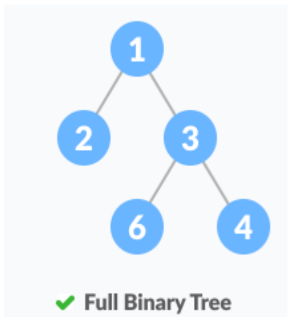
```
Input: Node 5
Output: Node 5 is at level 2.

Input: Node 10
Output: Node 10 is at level 4.

Input: Node 4
Output: Node 4 is at level 0.
```

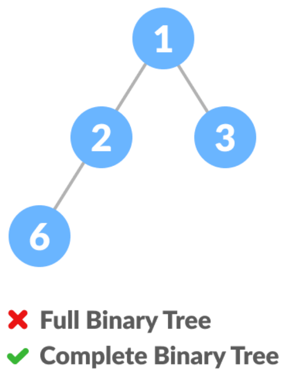
9) Full Binary tree:

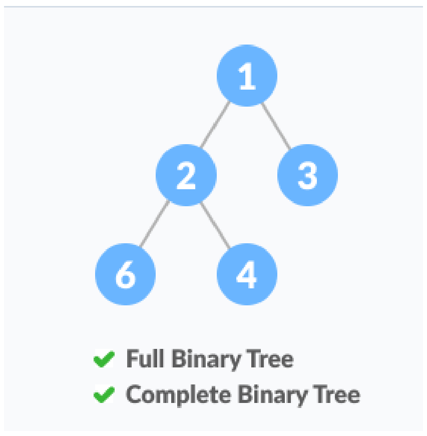
✓ A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.



10) Complete Binary tree:

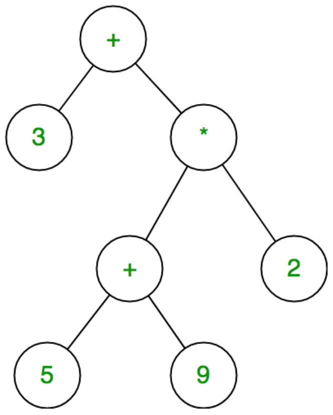
✓ A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.





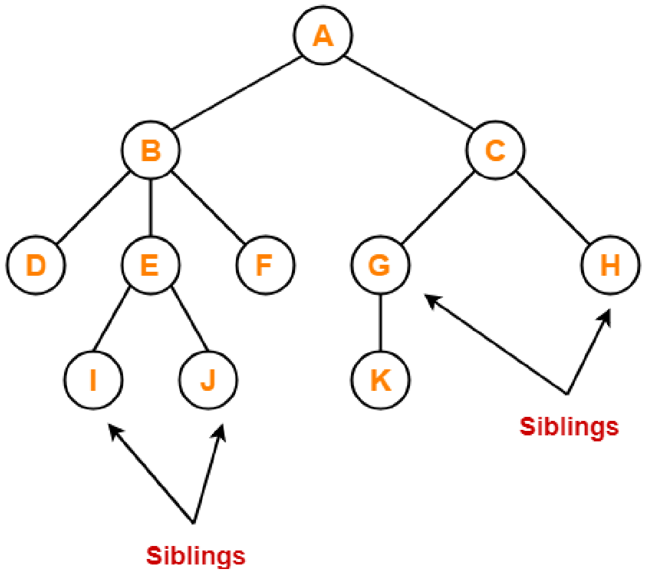
11) Expression Tree:

✓ Expression tree is a binary tree in which each internal node corresponds to operator and each leaf node corresponds to operand so for example expression tree for 3 + ((5+9)*2) would be:

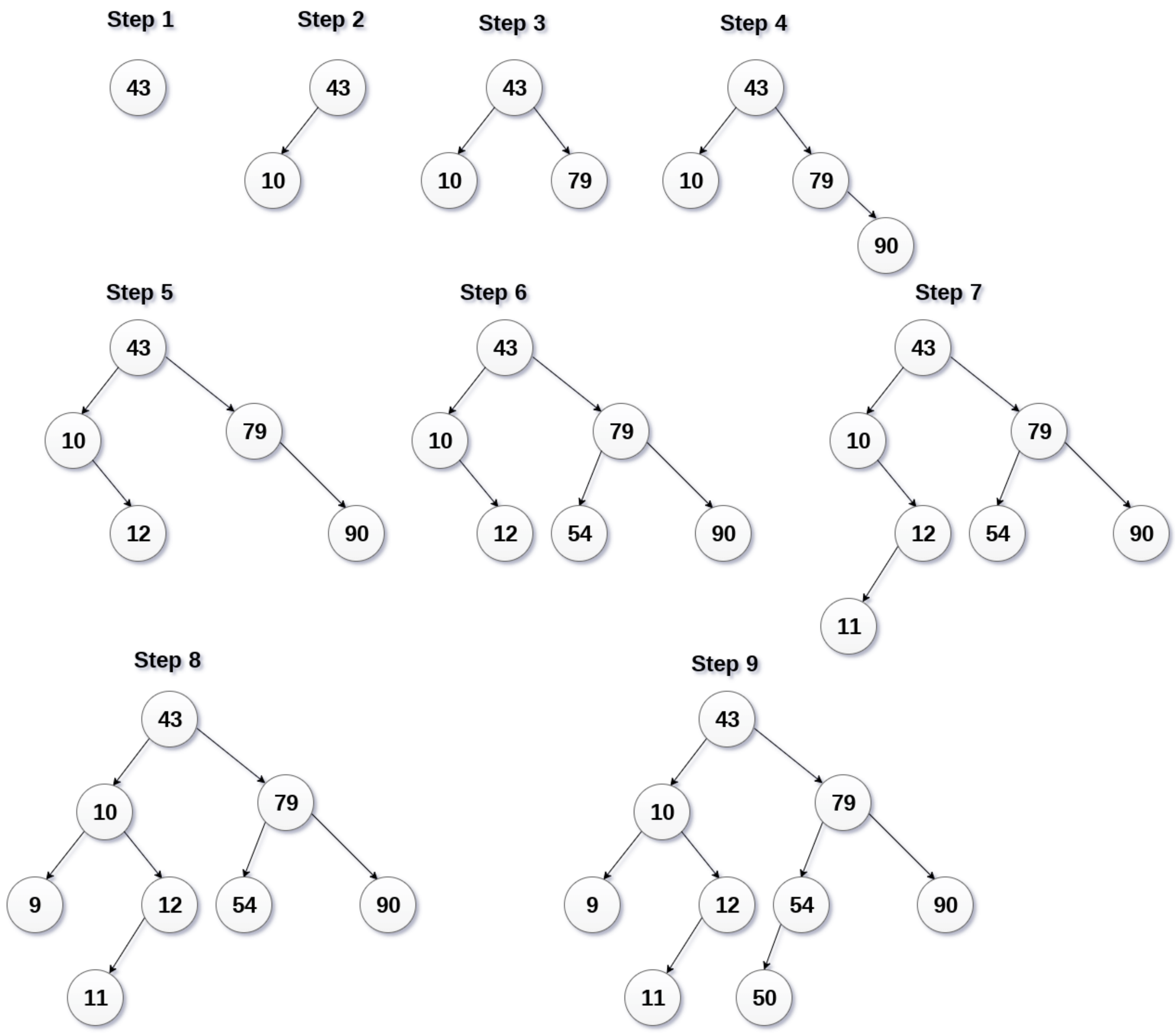


12) Sibling:

✓ Two nodes are said to be **siblings** if they are present at the same level, and their parents are same.



Q.2) Create the binary search tree using the following data elements.



Binary search Tree Creation

Q.3) Create the Heap tree / Heap Sort using the following data elements.

29,21,16,4,41,49,32,24,74

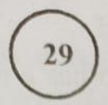


Fig. 7.8.1(a)

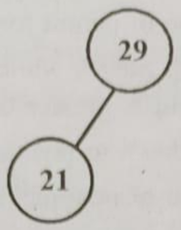


Fig. 7.8.1(b)

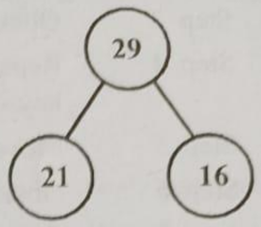


Fig. 7.8.1(c)

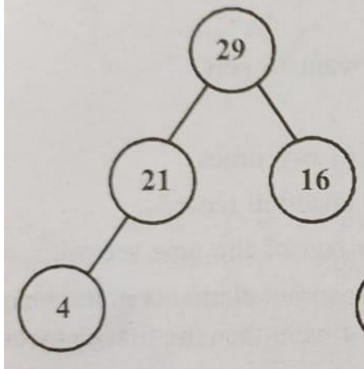


Fig. 7.8.1(d)

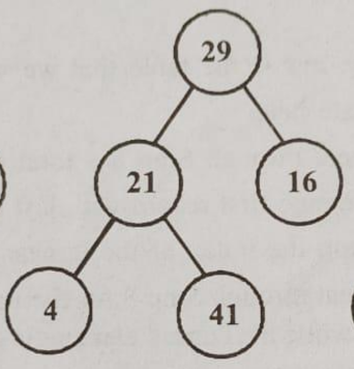


Fig. 7.8.1(e)

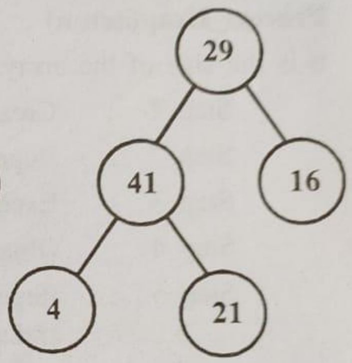


Fig. 7.8.1(f)

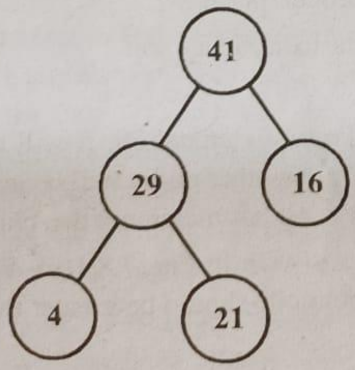


Fig. 7.8.1(g)

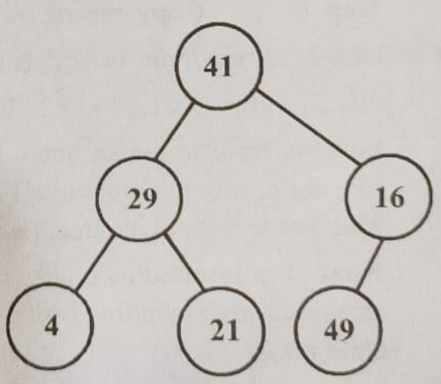


Fig. 7.8.1(h)

Insert 32 as a right child of 41as shown in Fig. 7.8.1(k).

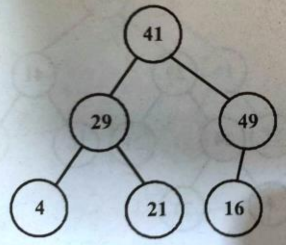


Fig. 7.8.1(i)

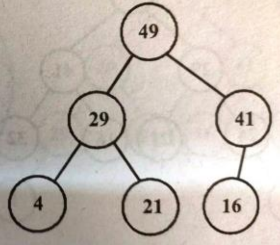


Fig. 7.8.1(j)

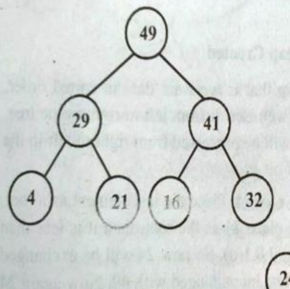


Fig. 7.8.1(k)

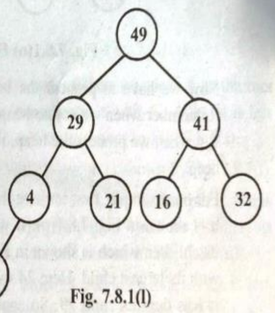


Fig. 7.8.1(l)

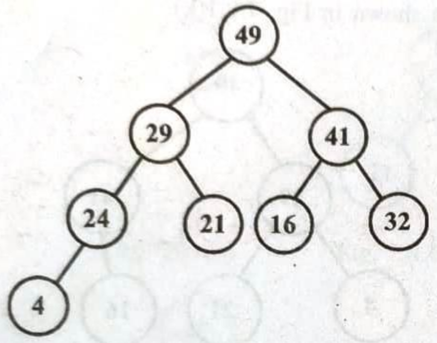


Fig. 7.8.1(m)

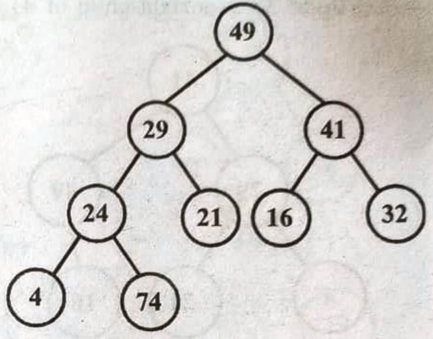


Fig. 7.8.1(n)

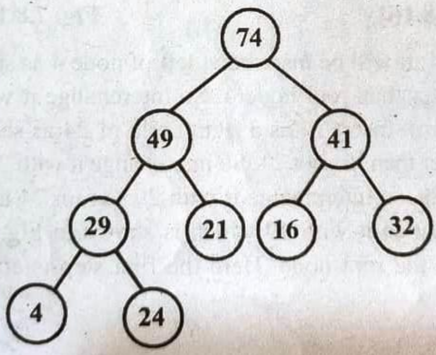
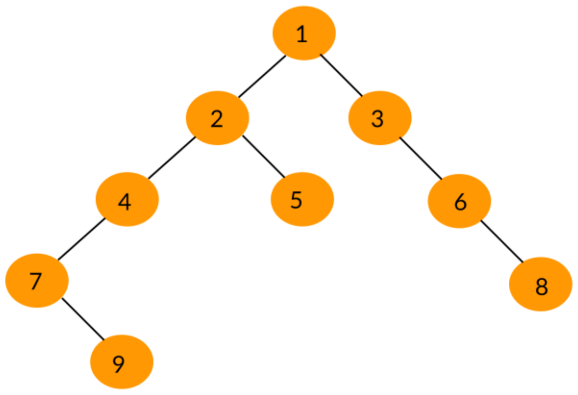


Fig. 7.8.1(o) Heap Created

Q.4) Using tree find out inorder, prorder, postorder (7)



Inorder Traversal: 7 9 4 2 5 1 3 6 8

Preorder Traversal: 1 2 4 7 9 5 3 6 8

Postorder Traversal: 9 7 4 5 2 8 6 3 1

OR

Q.4) Generate Postorder traversal of Tree from Inorder and Preorder traversal of tree without generating Tree.

Input:
In-order traversal inorder {4, 2, 5, 1, 3, 6}
Pre-order traversal preorder {1, 2, 4, 5, 3, 6}

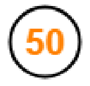
Output:

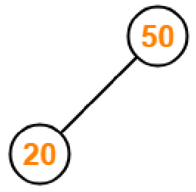
```
      1
     / \
    2   3
   / \   \
  4   5   6
```

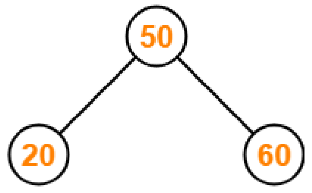
Post-order traversal is {4, 5, 2, 6, 3, 1}

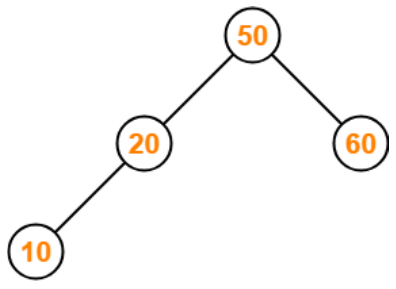
Q.5) Create the AVL tree using the following data elements.

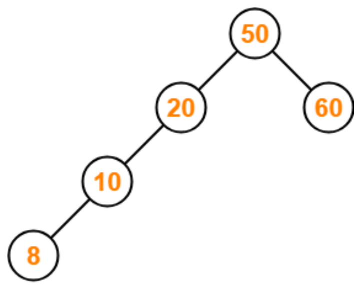
50 , 20 , 60 , 10 , 8 , 15 , 32 , 46 , 11 , 48

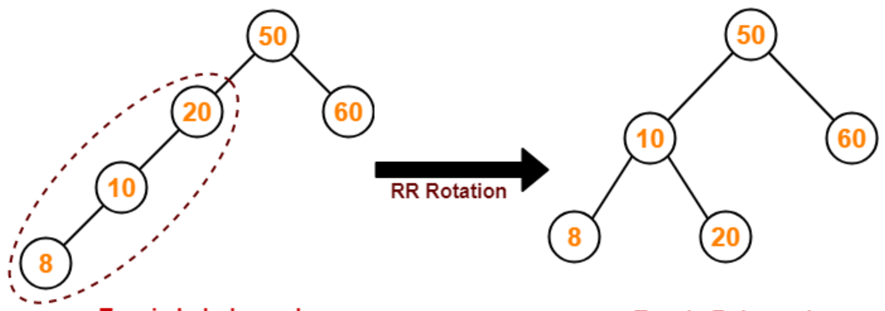

Tree is Balanced


Tree is Balanced


Tree is Balanced

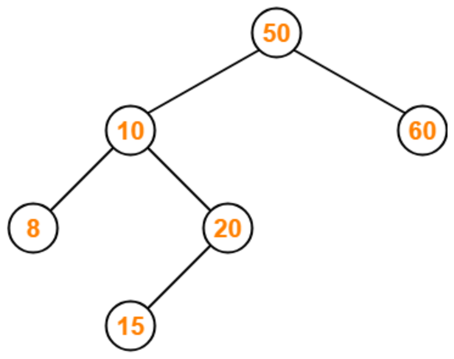

Tree is Balanced

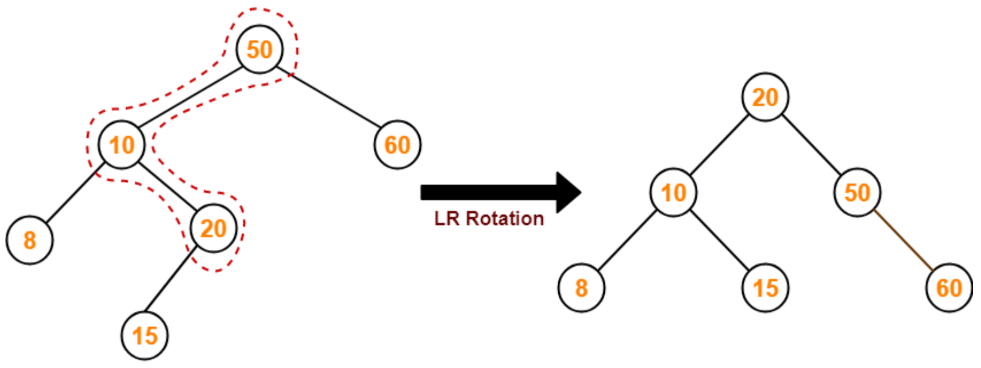

Tree is Imbalanced



Tree is Imbalanced

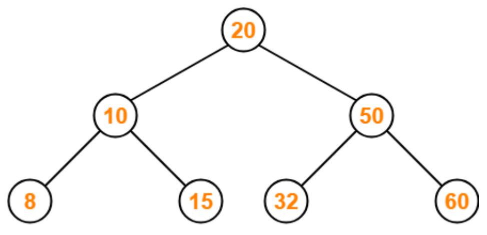
Tree is Balanced


Tree is Imbalanced

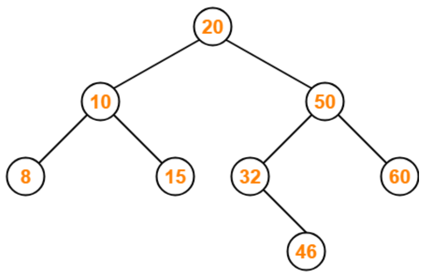


Tree is Imbalanced

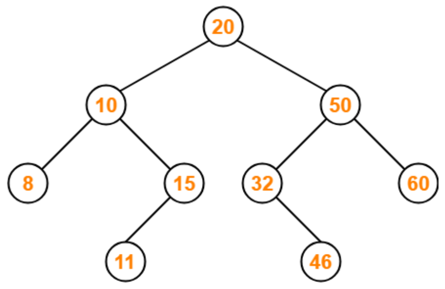
Tree is Balanced



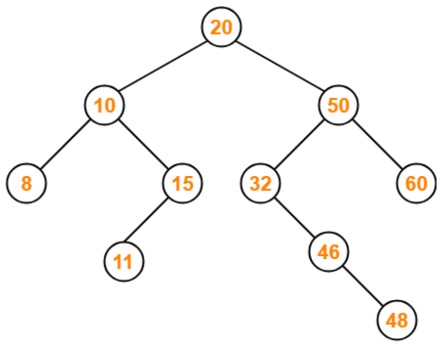
Tree is Balanced



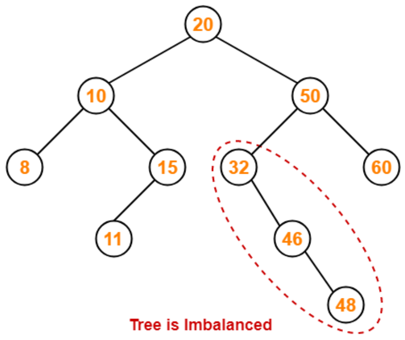
Tree is Balanced



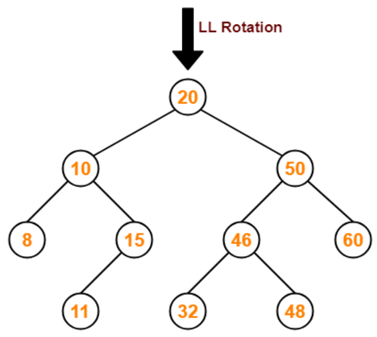
Tree is Balanced



Tree is imbalanced



Tree is Imbalanced

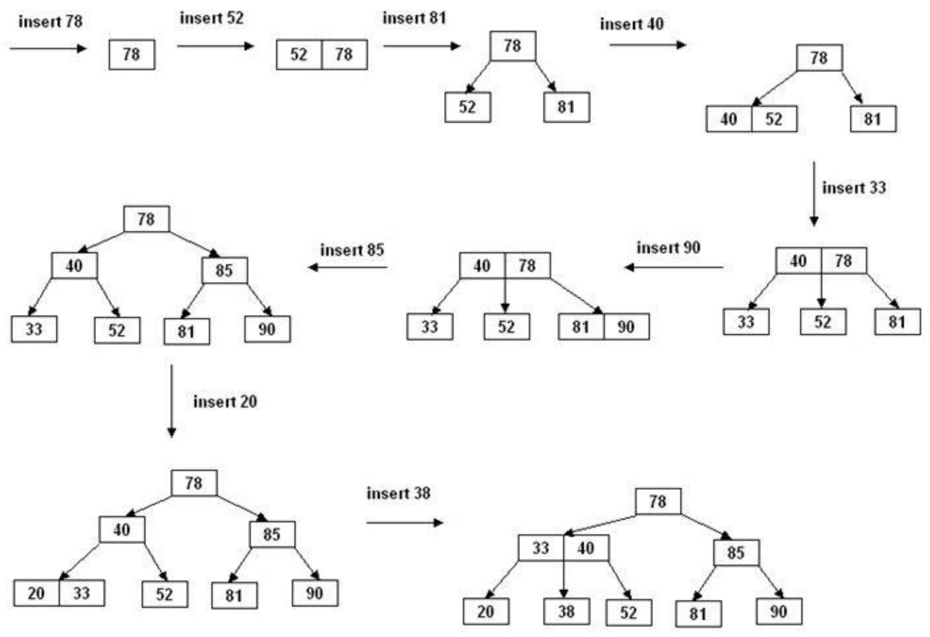


Tree is Balanced

Q.6) Create the B tree using the following data elements.

Insertion in B-Trees

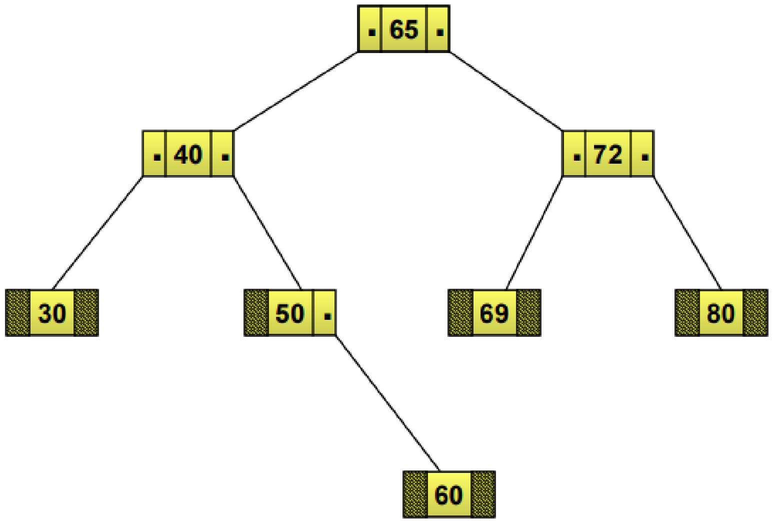
- **Insertion in a B-tree of odd order**
- Example: Insert the keys 78, 52, 81, 40, 33, 90, 85, 20, and 38 in this order in an initially empty B-tree of order 3



Q.7)Write a short note Threaded binary tree with example.

- ✓ A binary tree can be represented by using array representation or linked list representation.
- ✓ When a binary tree is represented using linked list representation.
- ✓ If any node is not having a child we use a NULL pointer.
- ✓ These special pointers are threaded and the binary tree having such pointers is called a threaded binary tree.
- ✓ Thread in a binary tree is represented by a dotted line.
- ✓ Consider the following binary search tree.

- ✓ Most of the nodes in this tree hold a NULL value in their left or right child fields.
- ✓ In this case, it would be good if these NULL fields are utilized for some other useful purpose

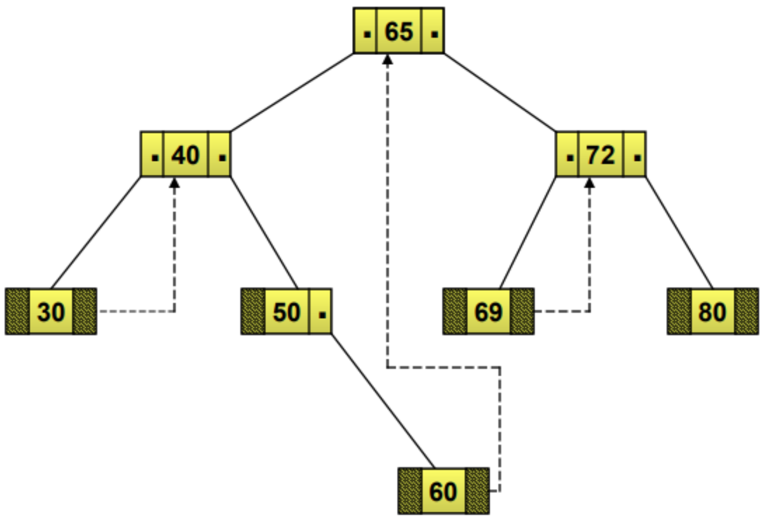


- ✓ In the threaded binary tree when there is only one thread is used then it is called as **one way threaded tree** and when both the threads are used then it is called the **two way threaded tree.**

❖ One Way Threaded Binary Trees

- ✓ The empty left child field of a node can be used to point to its inorder predecessor.
- ✓ Similarly, the empty right child field of a node can be used to point to its in-order successor.
- ✓ Such a type of binary tree is known as a one way threaded binary tree.

In-order :- 30 40 50 60 65 69 72 80

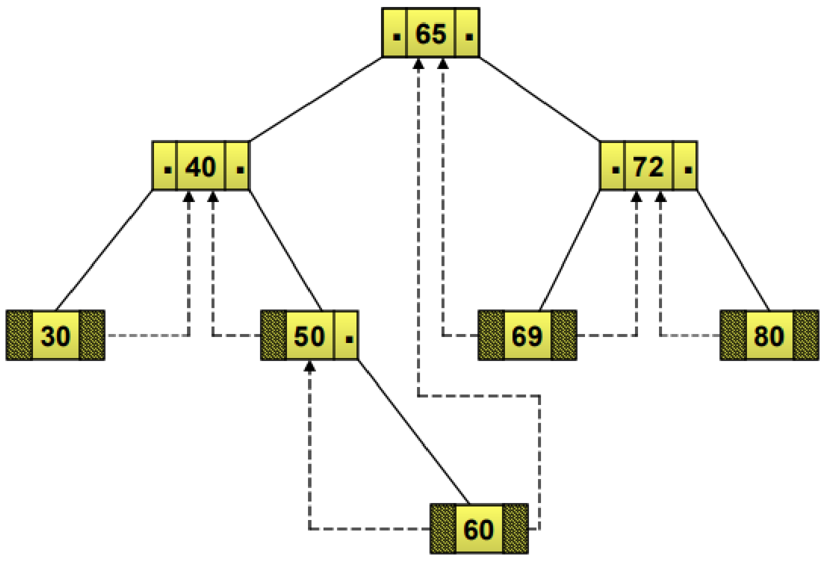


❖ **Two way Threaded Binary Trees:**

✓ A field that holds the address of its inorder successor or predecessor is known as thread.

✓ The empty left child field of a node can be used to point to its inorder predecessor.

Inorder :- 30 40 50 60 65 69 72 80

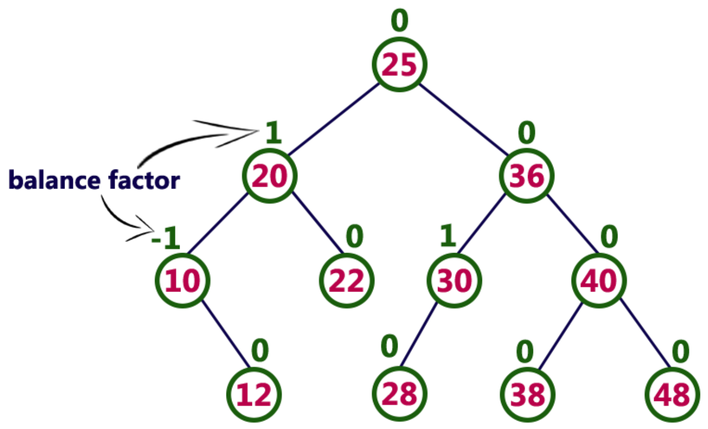


Q.8) Explain AVL Tree(Height Balance Tree) with example.

- ✓ AVL tree is a height-balanced binary search tree.
- ✓ That means, an AVL tree is also a binary search tree but it is a balanced tree.
- ✓ A binary tree is said to be balanced if, the difference between the heights of left and right subtrees of every node in the tree is either -1, 0 or +1.
- ✓ In an AVL tree, every node maintains an extra information known as balance factor.

- ✓ Balance factor of a node is the difference between the heights of the left and right subtrees of that node.
- ✓ The balance factor of a node is calculated either height of left subtree - height of right subtree (OR) height of right subtree - height of left subtree.

Balance factor = heightOfLeftSubtree – heightOfRightSubtree

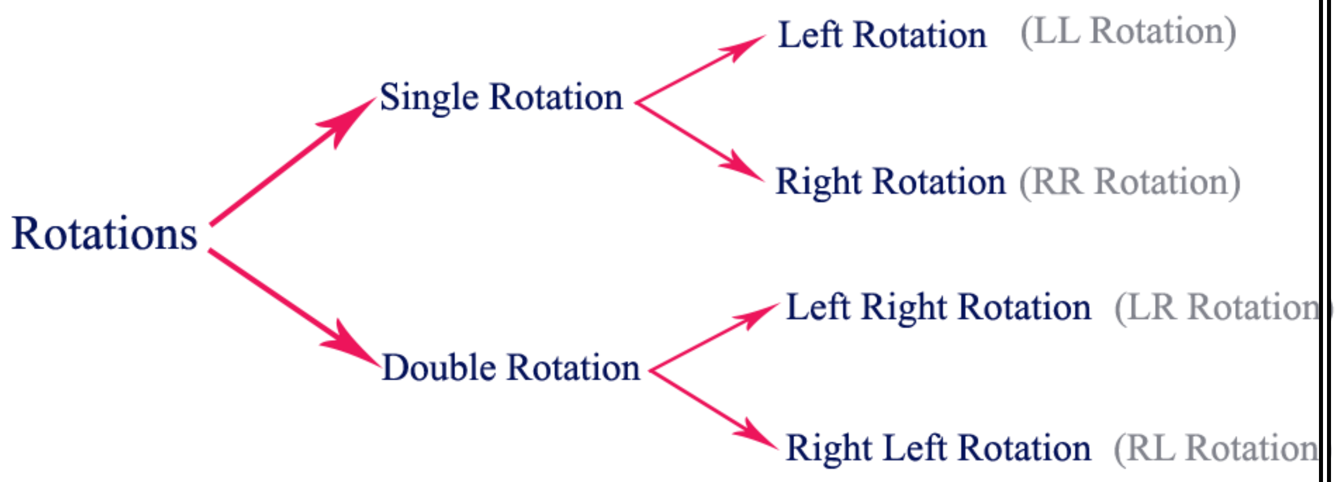


- ✓ The above tree is a binary search tree and every node is satisfying balance factor condition. So this tree is said to be an **AVL tree.**

❖ **<u>AVL Tree Rotations</u>**

- ✓ Whenever the tree becomes imbalanced due to any operation we use rotation operations to make the tree balanced.

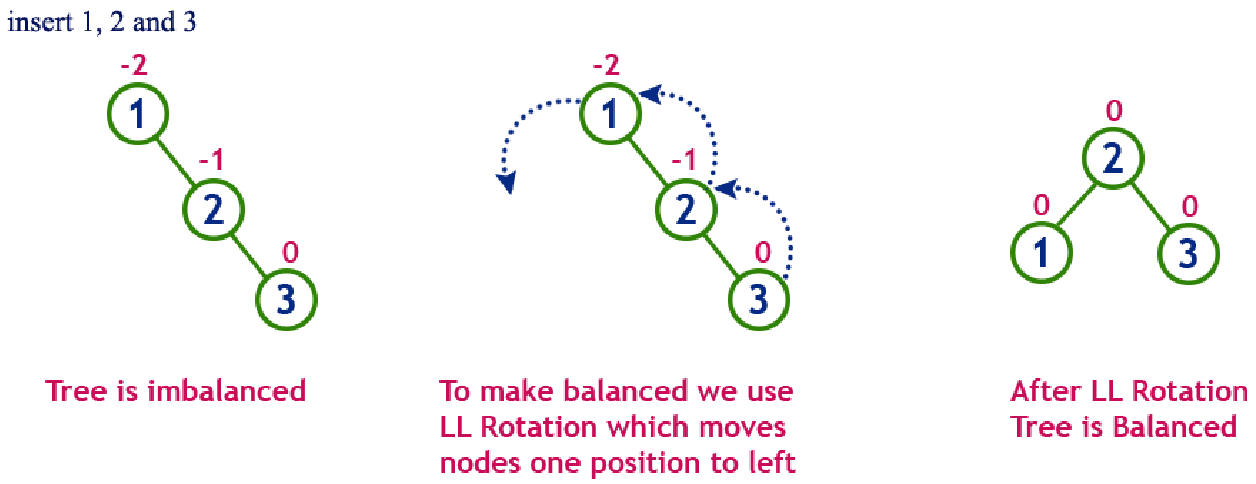
✓ **There are four rotations and they are classified** into **two types.**


Rotations
Single Rotation
Double Rotation
Left Rotation (LL Rotation)
Right Rotation (RR Rotation)
Left Right Rotation (LR Rotation
Right Left Rotation (RL Rotation

1) Single Left Rotation (LL Rotation)

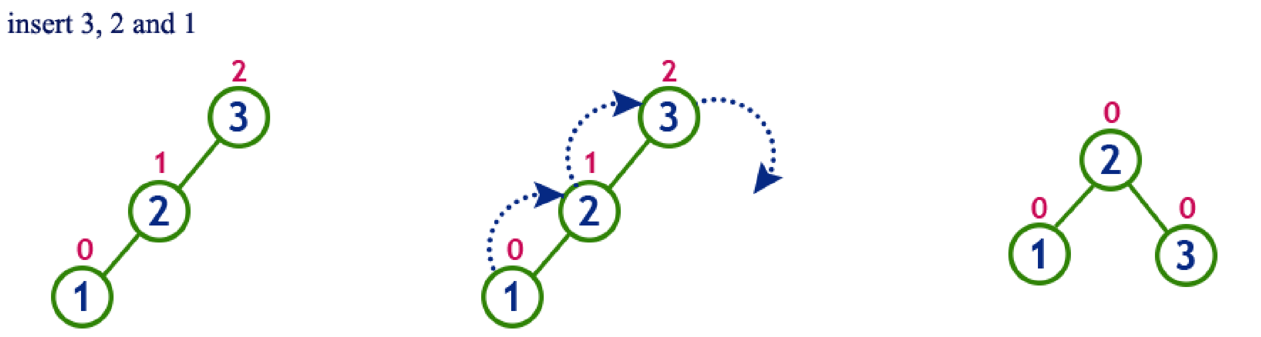
✓ In LL Rotation, every node moves one position to left from the current position.

✓ To understand LL Rotation, let us consider the following insertion operation in AVL Tree...


insert 1, 2 and 3
-2
1
-1
2
0
3
Tree is imbalanced
-2
1
-1
2
0
3
To make balanced we use LL Rotation which moves nodes one position to left
0
2
0
1
0
3
After LL Rotation Tree is Balanced

2) Single Right Rotation (RR Rotation)

- ✓ In RR Rotation, every node moves one position to right from the current position.
- ✓ To understand RR Rotation, let us consider the following insertion operation in AVL Tree...



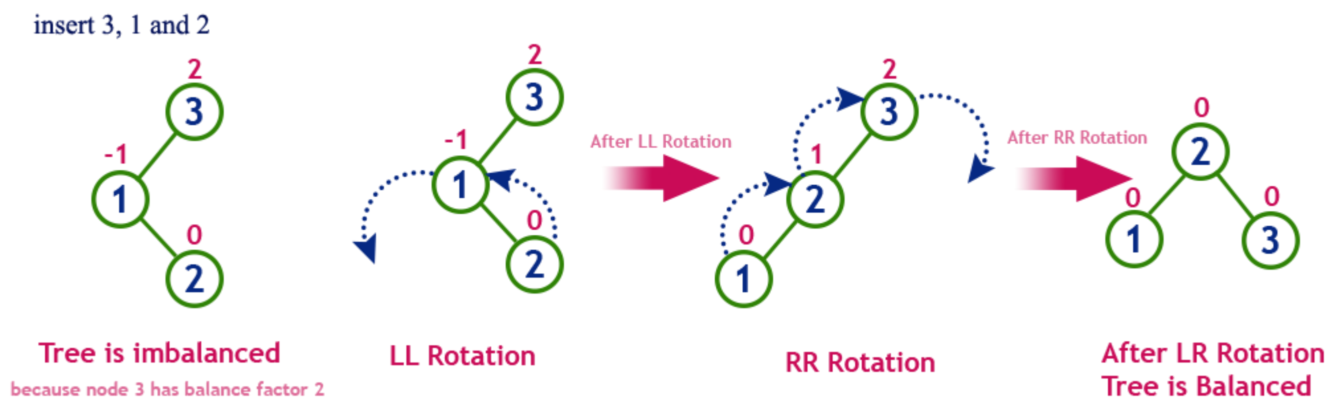
Tree is imbalanced
because node 3 has balance factor 2

To make balanced we use RR Rotation which moves nodes one position to right

After RR Rotation Tree is Balanced

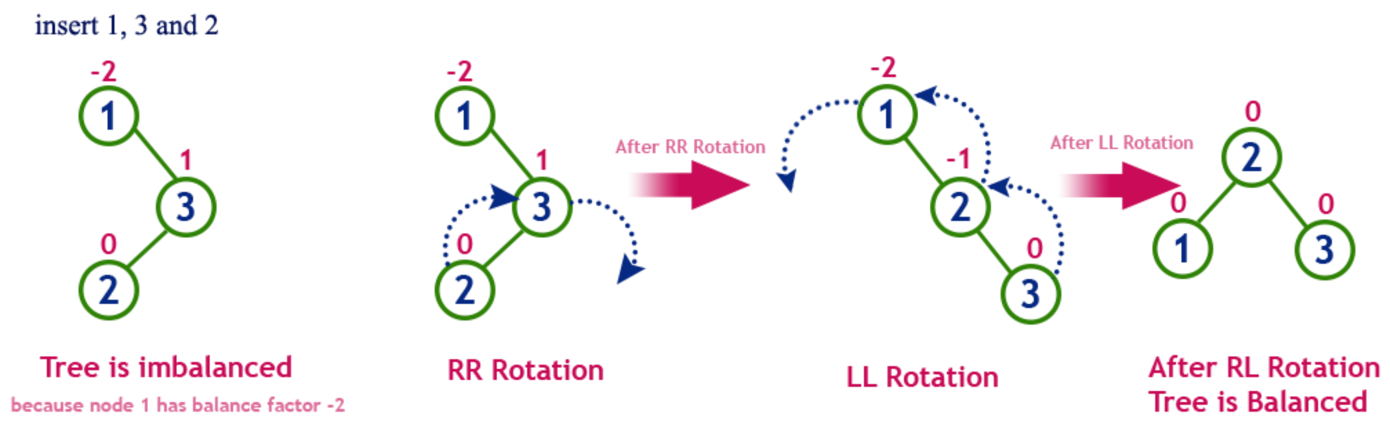
3) Left Right Rotation (LR Rotation)

- ✓ The LR Rotation is a sequence of single left rotation followed by a single right rotation.
- ✓ In LR Rotation, at first, every node moves one position to the left and one position to right from the current position.
- ✓ To understand LR Rotation, let us consider the following insertion operation in AVL Tree...



4) Right Left Rotation (RL Rotation)

- ✓ The RL Rotation is sequence of single right rotation followed by single left rotation.
- ✓ In RL Rotation, at first every node moves one position to right and one position to left from the current position.
- ✓ To understand RL Rotation, let us consider the following insertion operation in AVL Tree...


insert 1, 3 and 2
Tree is imbalanced
because node 1 has balance factor -2
RR Rotation
After RR Rotation
LL Rotation
After LL Rotation
After RL Rotation
Tree is Balanced