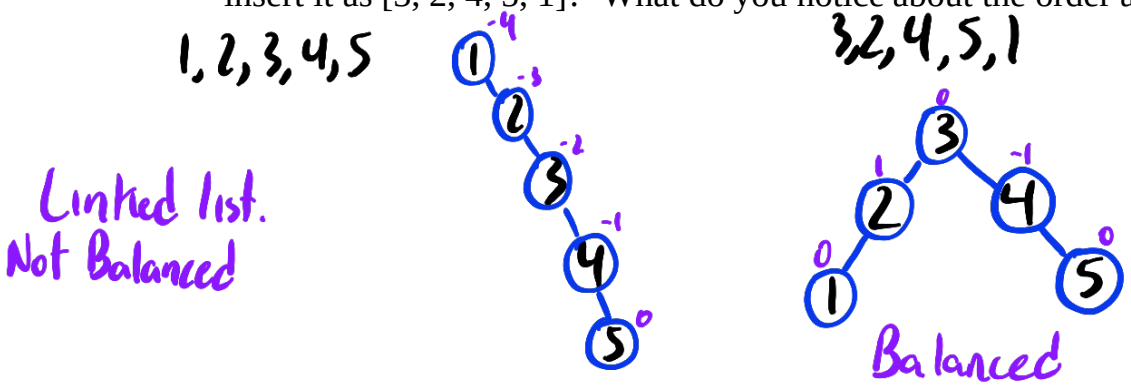
Wilson Liang

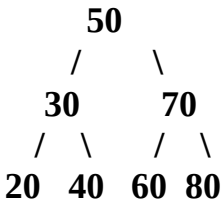
# Recitation12

1. A binary search tree is a tree in which for each node, all the nodes in the left sub-tree are smaller than the node and all the elements in the right sub-tree are larger than the node. Using a binary search tree of integers:

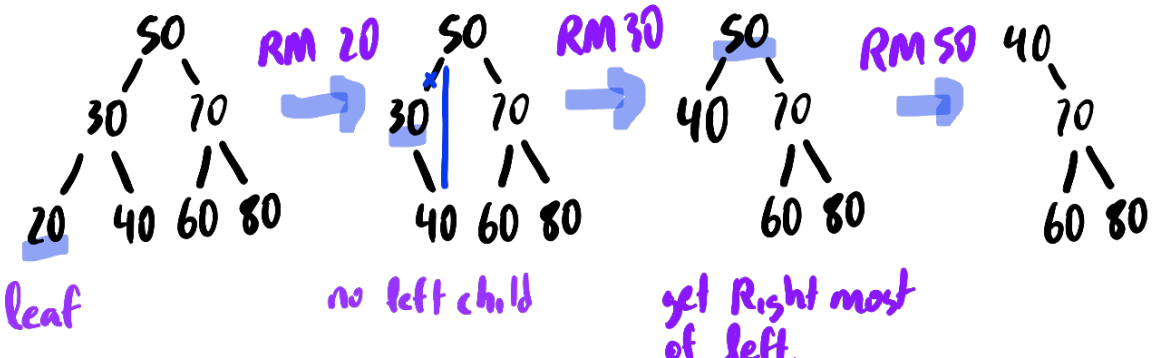
   a. Draw the binary search tree created by inserting these integers in the following order. [1, 2, 3, 4, 5]. What do you notice about this tree? What happens if you insert it as [3, 2, 4, 5, 1]? What do you notice about the order and the tree?




   b. You are given a binary search tree.




   Delete the following nodes one after another in the following order – 20, 30, 50




2. Write an iterative method to find the minimum value in a binary search tree. This method will return the lowest integer value in the tree. When the function

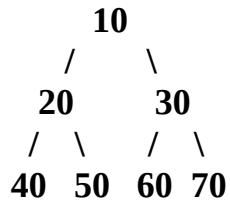
is called, the root of the tree is passed as the argument to the function and it returns the integer value of the smallest element in the binary search tree. You may assume that the nodes have all necessary access and modifier methods.

```
public int minValue(Node root)
  If(root ==null) throw new NullPointerException();
  Node ptr =root;
  while (ptr.getLeft() != null)
     ptr = ptr.getleftData();
  return ptr.getData();
```

3. Write a method to print the nodes of a binary tree in level order:

Breath First Search → Queue
- Traversal level by level

Input:




Output: {10, 20, 30, 40, 50, 60, 70}
Assume TreeNode class is defined with fields: data (int), left (TreeNode), and right (TreeNode).
Write a method `levelOrder(TreeNode root)` that print the nodes of a binary tree in level order. **Hint**: use Queue.

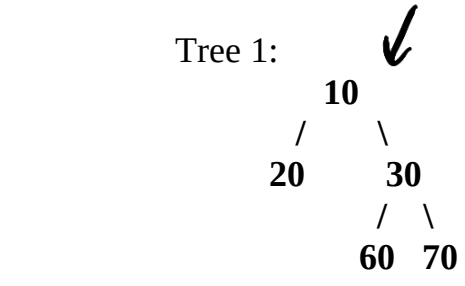
```
void levelOrder(TreeNode root)
   Queue<TreeNode> q = new LinkedList<>();
   q.enqueue(root)
   while(!q.isEmpty())
      TreeNode temp = q.remove()
      system.out.println(temp.data + " ");
```

```
If (temp.left != Null)
    q.add(temp.left)
If (temp.right != Null)
    q.add(temp.right)
```

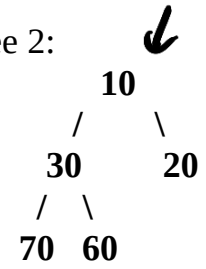
4. Check if two trees are mirror images of each other. Look at the example input below for better understanding.

Input:

Tree 1:




Tree 2:




```
public boolean areMirror(TNode node1, TNode node2)
    If(node1==null && node2==null)
        return true;
    If(node1==null || node2==null)
        return false;
    If(node1.data != node2.data)
        return false
    return
        isMirror(node1.left,node2.right)
    && isMirror(node1.right,node2.left)
```

Output: **True**

Write a method areMirror (TreeNode node1, TreeNode node2) where you are given the root of both the trees as input. It should return **True** if the trees are mirror images else return **False.**