**Solution @30-10-23**

**206. Reverse Linked List**

```
class ListNode {
    int val;
    ListNode next;

    public ListNode(int x) {
        val = x;
    }
}

public class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode newHead = null;
        while (head != null) {
            ListNode current = head;
            head = head.next;
            current.next = newHead;
            newHead = current;
        }
        return newHead;
    }

    public static String listToString(ListNode head) {
        StringBuilder sb = new StringBuilder();
        while (head != null) {
            sb.append(head.val);
            sb.append(" -> ");
            head = head.next;
        }
        sb.append("null");
```

```
        return sb.toString();
    }

    public static void main(String[] args) {
        int[] n1 = { 1, 2, 3, 4, 5 };

        ListNode l1 = new ListNode(n1[0]);
        ListNode h1 = l1;
        for (int i = 1; i < n1.length; i++) {
            ListNode n = new ListNode(n1[i]);
            l1.next = n;
            l1 = l1.next;
        }

        Solution c = new Solution();
        h1 = c.reverseList(h1);

        String reversedList = listToString(h1);
        System.out.println(reversedList);
    }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java1> javac Solution.java
PS C:\Users\Ajeet\Desktop\java1> java Solution
5 -> 4 -> 3 -> 2 -> 1 -> null
```

## 199. Binary Tree right side View

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;
```

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int val) {
        this.val = val;
    }
}

public class Solution {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) {
            return result;
        }

        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);

        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = queue.poll();
                if (i == size - 1) {
                    result.add(node.val);
                }
                if (node.left != null) {
                    queue.offer(node.left);
                }
```

```
            if (node.right != null) {
                queue.offer(node.right);
            }
        }
    }

    return result;
  }

  public static void main(String[] args) {
    // Create a sample binary tree
    TreeNode root = new TreeNode(1);
    root.left = new TreeNode(2);
    root.right = new TreeNode(3);
    root.left.right = new TreeNode(5);
    root.right.right = new TreeNode(4);

    Solution solution = new Solution();
    List<Integer> rightView = solution.rightSideView(root);

    // Print the right side view
    System.out.println(rightView);
  }
}
```

```
PS C:\Users\Ajeet\Desktop\java1> javac Solution.java
PS C:\Users\Ajeet\Desktop\java1> java Solution
[1, 3, 4]
```