Daily Practice Problem ---Date 30/10/2023

Problem-1- Reverse Linked List

```
package ishwarchavan.com;
import java.util.LinkedList;

public class ReverseLinked{      //class created

	static Node head;

	static class Node {     //static class created

		int data;      //variable created
		Node next;

		Node(int d) {
			data = d;      //assigning the value
			next = null;
		}
	}

	Node reverse(Node node) {   //reverse function created
		Node prev = null;    //assigning null value
		Node current = node;
		Node next = null;

		while (current != null) {     //checking condition if the true then execute
below statement
			next = current.next;
			current.next = prev;
			prev = current;
			current = next;
		}
		node = prev;    //assigning prev value
		return node;
	}

	void printList(Node node) {    // prints content of double linked list
		while (node != null) {    //checking condition
			System.out.print(node.data + " ");
			node = node.next;
		}
	}

	public static void main(String[] args) {     //main program started
		ReverseLinked list = new ReverseLinked ();   //object  created
		list.head = new Node(1);
		list.head.next = new Node(2);     //nodes created
		list.head.next.next = new Node(3);
		list.head.next.next.next = new Node(4);
		list.head.next.next.next.next = new Node(5);

		System.out.println("Given Linked list");   //printing statement
		list.printList(head);
		head = list.reverse(head);
		System.out.println("");
		System.out.println("Reversed linked list ");
		list.printList(head);
	}
}
```

Problem-2 - Right view of binary tree

```
package ishwarchavan.com;

class Node {        //A binary tree node

	int data;
	Node left, right;

	Node(int item)
	{
		data = item;    //assigning value
		left = right = null;
	}
}

class Max_level {          //class to access maximum level by reference

	int max_level;
}

 public class RightBinaryView {

	Node root;
	Max_level max = new Max_level();

	void rightViewUtil(Node node, int level,      // Recursive function to print
right view of a binary
				Max_level max_level)
	{

		if (node == null)    //checking condition
			return;

		if (max_level.max_level < level) {    // If this is the last Node of its
level
			System.out.print(node.data + " ");
			max_level.max_level = level;
		}

		rightViewUtil(node.right, level + 1, max_level);     // Recur for right
subtree first, then left subtree
		rightViewUtil(node.left, level + 1, max_level);
	}

	void rightView() { rightView(root); }

	void rightView(Node node)
	{

		rightViewUtil(node, 1, max);
	}
	public static void main(String args[])       //main program started
	{
		RightBinaryView tree = new RightBinaryView();
		tree.root = new Node(1);
		tree.root.left = new Node(2);
		tree.root.right = new Node(3);
		tree.root.left.left = new Node(4);
		tree.root.left.right = new Node(5);
		tree.root.right.left = new Node(6);
		tree.root.right.right = new Node(7);
		tree.root.right.left.right = new Node(8);

		tree.rightView();    //calling function
	}
}
```