1. Program for Merge Two 2D Arrays by Summing Values

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

public class MergeTwoArrays {

    public static int[][] mergeArrays(int[][] nums1, int[][] nums2) {
        int n = nums1.length;
        int m = nums2.length;
        List<int[]> res = new ArrayList<>(); // List to store the merged pairs

        int i = 0, j = 0;

        // Merge arrays using two-pointer approach
        while (i < n && j < m) {
            if (nums1[i][0] == nums2[j][0]) {
                // If integers are equal, merge with summed frequency
                res.add(new int[]{nums1[i][0], nums1[i][1] + nums2[j][1]});
                i++; j++;
            } else if (nums1[i][0] < nums2[j][0]) {
                // Add nums1 pair since its integer is smaller
                res.add(new int[]{nums1[i][0], nums1[i][1]});
                i++;
            } else {
                // Add nums2 pair since its integer is smaller
                res.add(new int[]{nums2[j][0], nums2[j][1]});
                j++;
            }
        }

        // Add remaining pairs from nums1
        while (i < n) {
            res.add(new int[]{nums1[i][0], nums1[i][1]});
            i++;
        }

        // Add remaining pairs from nums2
        while (j < m) {
            res.add(new int[]{nums2[j][0], nums2[j][1]});
            j++;
        }

        // Convert List of pairs to a 2D array
        int s = res.size();
        int[][] mergedArray = new int[s][2];
        for (int l = 0; l < s; l++) {
            mergedArray[l] = res.get(l);
        }

        return mergedArray;
    }
    public static void main(String[] args) {
```

```
        int[][] nums1 = {{1, 3},{2, 4}, {5, 6}};

        int[][] nums2 = {{2, 5}, {3, 2}, {5, 8}};

        int[][] numMarge = mergeArrays(nums1, nums2);

        System.out.println("Merged Array:");
        for (int[] pair : numMarge) {
            System.out.print("{"+pair[0] + "," + pair[1]+"} ");
        }
    }
}
```

OUTPUT

Merged Array:

{1,3} {2,9} {3,2} {5,14}

2. Program for Maximum Width of Binary Tree

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

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int val) {
        this.val = val;
    }
}

public class MaximumWidthBinaryTree {

    public static int maxWidthOfBinaryTree(TreeNode root) {
        if (root == null) {
            return 0;
        }

        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        int maxWidth = 0;

        while (!queue.isEmpty()) {
            int size = queue.size();
            maxWidth = Math.max(maxWidth, size);

            for (int i = 0; i < size; i++) {
                TreeNode node = queue.poll();
                if (node.left != null) {
                    queue.offer(node.left);
                }
                if (node.right != null) {
                    queue.offer(node.right);
                }
            }
        }

        return maxWidth;
    }

    public static void main(String[] args) {
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(3);
        root.right = new TreeNode(2);
        root.left.left = new TreeNode(5);
        root.left.right = new TreeNode(3);
        root.right.right = new TreeNode(9);

        int maxWidth = maxWidthOfBinaryTree(root);
```

```
        System.out.println("Maximum Width of Binary Tree: " + maxWidth);
    }
}
```

OUTPUT

Maximum Width of Binary Tree: 3