

PROBLEM SHEET 1 solution 1:

The screenshot shows a programming environment with the following details:

- Header:** Three 90 Ending, Courses, Tutorials, Practice, Jobs.
- Left Sidebar:** Problem, Editorial, Submissions, Comments.
- Output Window:** Compiling with Custom Input, Y.O.G.I. (AI Bot).
- Code Editor:** Java (21) code for reversing an array.
- Results Summary:**
 - Test Cases Passed: 1115 / 1115
 - Attempts : Correct / Total: 1 / 1 (Accuracy: 100%)
 - Points Scored: 2 / 2
 - Time Taken: 0.84
- Buttons:** Solve Next, Custom Input, Compile & Run, Submit.

Solution 2:

The screenshot shows a programming environment with the following details:

- Header:** Three 90 Ending, Courses, Tutorials, Practice, Jobs.
- Left Sidebar:** Problem, Editorial, Submissions, Comments.
- Output Window:** Compiling with Custom Input, Y.O.G.I. (AI Bot).
- Code Editor:** Java (21) code for finding min and max in an array.
- Results Summary:**
 - Test Cases Passed: 1111 / 1111
 - Attempts : Correct / Total: 1 / 1 (Accuracy: 100%)
 - Points Scored: 1 / 1
 - Time Taken: 0.29
- Buttons:** Solve Next, Custom Input, Compile & Run, Submit.

Solution 3:

The screenshot shows a programming environment with the following details:

- Header:** Three 90 Ending, Courses, Tutorials, Practice, Jobs.
- Left Sidebar:** Problem, Editorial, Submissions, Comments.
- Output Window:** Compiling with Custom Input, Y.O.G.I. (AI Bot).
- Code Editor:** Java (21) code for finding the kth smallest element using quickselect.
- Results Summary:**
 - Test Cases Passed: 1121 / 1121
 - Attempts : Correct / Total: 1 / 2 (Accuracy: 50%)
 - Points Scored: 4 / 4
 - Time Taken: 0.57
- Buttons:** Solve Next, Custom Input, Compile & Run, Submit.

Solution 4:

A screenshot of a LeetCode problem page. The title is "Three Sum". The code editor shows a Java solution for finding the union of two arrays. The code uses a HashSet to store unique elements from both arrays and then converts it back to an ArrayList. The submission was successful, passing all test cases (1111 / 1111), with an accuracy of 100% and a time taken of 1.01 seconds. Points scored were 2/2, totaling a score of 9.

```
1 import java.util.*;
2
3 class Solution {
4     public ArrayList<Integer> findUnion(int a[], int b[]) {
5
6         // HashSet unique elements store karta hai
7         HashSet<Integer> set = new HashSet<>();
8
9         // Add elements of first array
10        for (int i = 0; i < a.length; i++) {
11            set.add(a[i]);
12        }
13
14        // Add elements of second array
15        for (int i = 0; i < b.length; i++) {
16            set.add(b[i]);
17        }
18
19        // Convert set to ArrayList
20        return new ArrayList<>(set);
21    }
22}
```

Solution 5:

A screenshot of a LeetCode problem page. The title is "Find Largest Value in Array". The code editor shows a Java solution for finding the largest value in an array. It initializes a variable max to the first element and iterates through the array, updating max whenever it finds a larger value. The submission was successful, passing all test cases (1115 / 1115), with an accuracy of 100% and a time taken of 0.79 seconds. Points scored were 1/1, totaling a score of 10.

```
1 class Solution {
2     public int largest(int[] arr) {
3
4         int max = arr[0];
5
6         for (int i = 1; i < arr.length; i++) {
7             if (arr[i] > max) {
8                 max = arr[i];
9             }
10        }
11
12        return max;
13    }
14}
```

Solution 6:

A screenshot of a LeetCode problem page. The title is "Rotate Array". The code editor shows a Java solution for rotating an array. It first stores the last element in a temporary variable last, then iterates from the end of the array towards the beginning, shifting each element to its predecessor's position. Finally, it places the last element at the first position. The submission was successful, passing all test cases (1115 / 1115), with an accuracy of 100% and a time taken of 1.09 seconds. Points scored were 1/1, totaling a score of 11.

```
1 class Solution {
2     public void rotate(int[] arr) {
3
4         int n = arr.length;
5
6         int last = arr[n - 1];
7
8         for (int i = n - 1; i > 0; i--) {
9             arr[i] = arr[i - 1];
10        }
11
12        arr[0] = last;
13    }
14}
```

Solution 7:

A screenshot of a LeetCode problem page for "Maximum Subarray" (Problem 53). The code editor shows the following Java solution:

```
1 class Solution {
2     int maxSubarraySum(int arr[]) {
3         int currentSum = arr[0];
4         int maxSum = arr[0];
5
6         for (int i = 1; i < arr.length; i++) {
7             currentSum = Math.max(arr[i], currentSum + arr[i]);
8             maxSum = Math.max(maxSum, currentSum);
9         }
10
11     }
12
13     return maxSum;
14 }
```

The "Output Window" shows "Problem Solved Successfully" with 1120 / 1120 test cases passed, 1 / 1 attempts correct, 100% accuracy, 4 / 4 points scored, and a time taken of 0.55. Buttons for "Custom Input", "Compile & Run", and "Submit" are visible.

Solution 8:

A screenshot of a LeetCode problem page for "Search Insert Position" (Problem 35). The code editor shows the following Java solution:

```
1 class Solution {
2     public int searchInsert(int[] nums, int target) {
3         int left = 0, right = nums.length - 1;
4
5         while (left <= right) {
6             int mid = left + (right - left) / 2;
7         }
8     }
9 }
```

The "Output Window" shows "Accepted" with 66 / 66 testcases passed by Ayush_Ishu7 at Feb 05, 2026 20:26. It includes runtime (0 ms, 100.00%), memory (45.04 MB, 9.28%), and a bar chart showing memory usage. The "Test Result" section shows "Accepted" with runtime 0 ms for Case 1, Case 2, and Case 3.

Solution 9:

A screenshot of a LeetCode problem page for "Reverse String" (Problem 151). The code editor shows the following Java solution:

```
1 import java.util.HashMap;
2
3 class Solution {
4     public int[] twoSum(int[] nums, int target) {
5         HashMap<Integer, Integer> map = new HashMap<>();
6
7         for (int i = 0; i < nums.length; i++) {
8             int complement = target - nums[i];
9
10            if (map.containsKey(complement)) {
11                return new int[] { map.get(complement), i };
12            }
13
14            map.put(nums[i], i);
15        }
16
17        return new int[] {};
18    }
19 }
```

The "Output Window" shows "Accepted" with 63 / 63 testcases passed by Ayush_Ishu7 at Feb 05, 2026 20:25. It includes runtime (2 ms, 99.15%), memory (46.93 MB, 76.86%), and a bar chart showing memory usage. The "Test Result" section shows "Accepted" with runtime 0 ms for Case 1, Case 2, and Case 3.

Solution 10:

The screenshot shows a programming environment with the following details:

- Header:** Get 90% Refund, Courses, Tutorials, Practice, Jobs.
- Toolbar:** Problem, Editorial, Submissions, Comments, Start Timer.
- Output Window:** Java (21) code editor.
- Compilation Results:** Custom Input, Y.O.G.I. (AI Bot).
- Problem Solved Successfully:** 1120 / 1120, 1 / 1, Accuracy: 100%.
- Points Scored:** 4 / 4.
- Time Taken:** 0.61.
- Code:** A Java solution for the problem.
- Buttons:** Custom Input, Compile & Run, Submit.

```
1 class Solution {
2     static int minJumps(int[] arr) {
3         int n = arr.length;
4
5         if (n == 1) return 0;
6
7         if (arr[0] == 0) return -1;
8
9         int maxReach = arr[0];
10        int steps = arr[0];
11        int jumps = 1;
12
13        for (int i = 1; i < n; i++) {
14            if (i == n - 1) return jumps;
15
16            maxReach = Math.max(maxReach, i + arr[i]);
17
18            steps--;
19
20            if (steps == 0) {
21                jumps++;
22
23                if (i >= maxReach) return -1;
24
25                steps = maxReach - i;
26            }
27        }
28    }
}
```