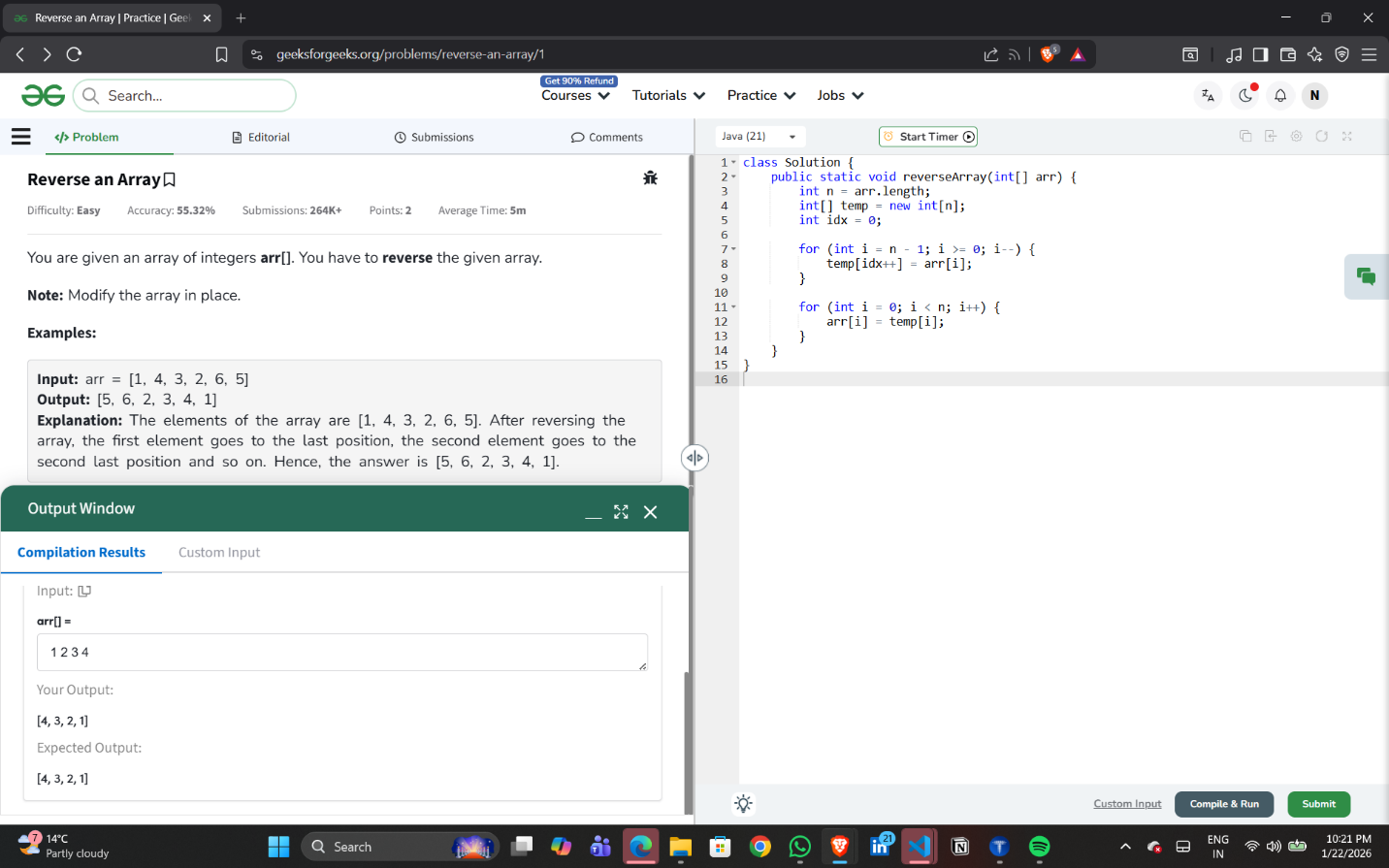
**Experiment 1 – Question 1**  
**Title:** Reverse an Array

**Algorithm:**

1. Create a temporary array of same size.
2. Copy elements from original array in reverse order.
3. Copy the temporary array back to the original array.

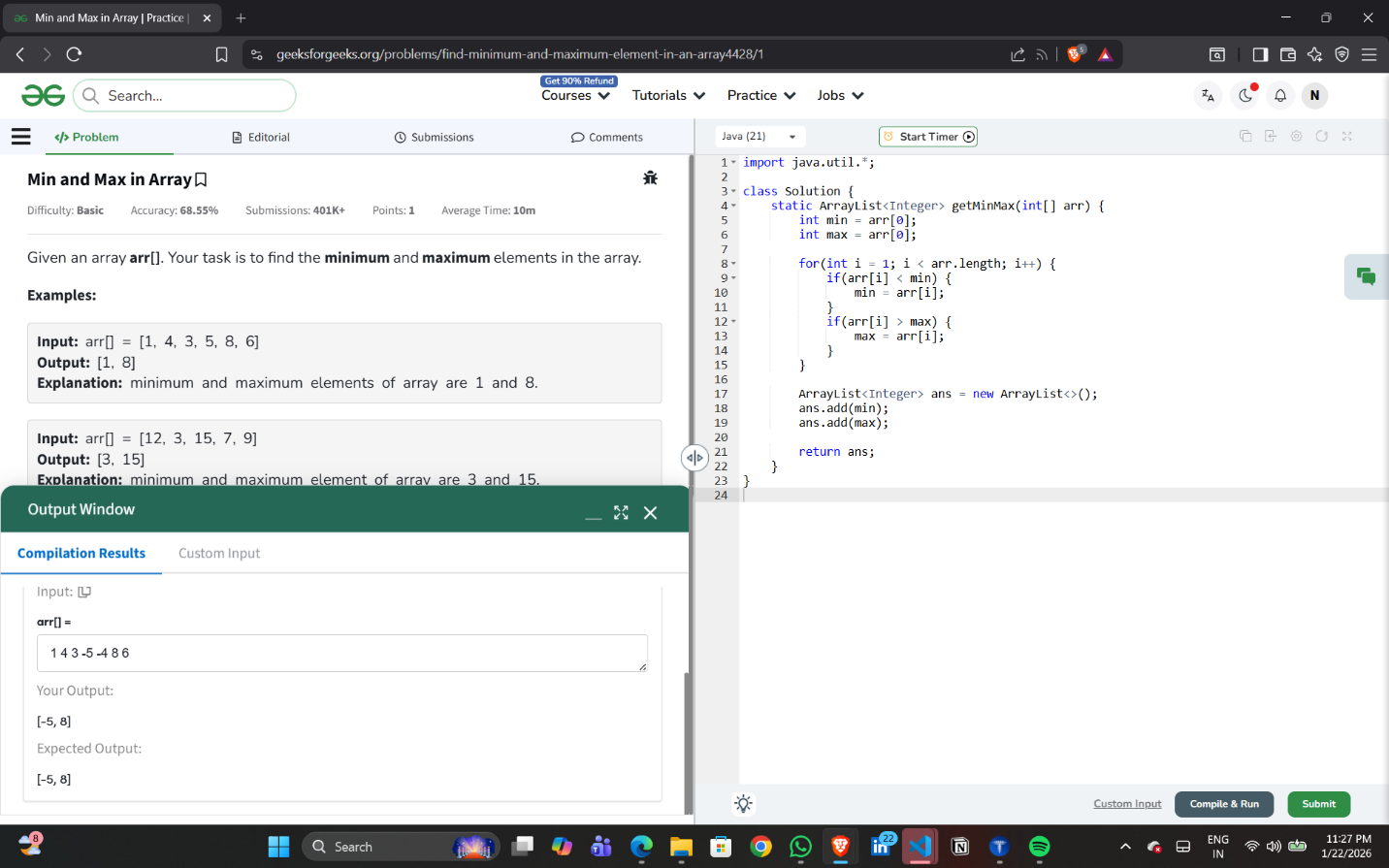
**Output:**  


**Experiment 1 – Question 2**  
**Title: Find Minimum and Maximum in an Array**

**Algorithm:**

1. **1. Initialize min and max as first element.**
2. **2. Traverse the array.**
3. **3. Update min and max.**
4. **4. Return result.**

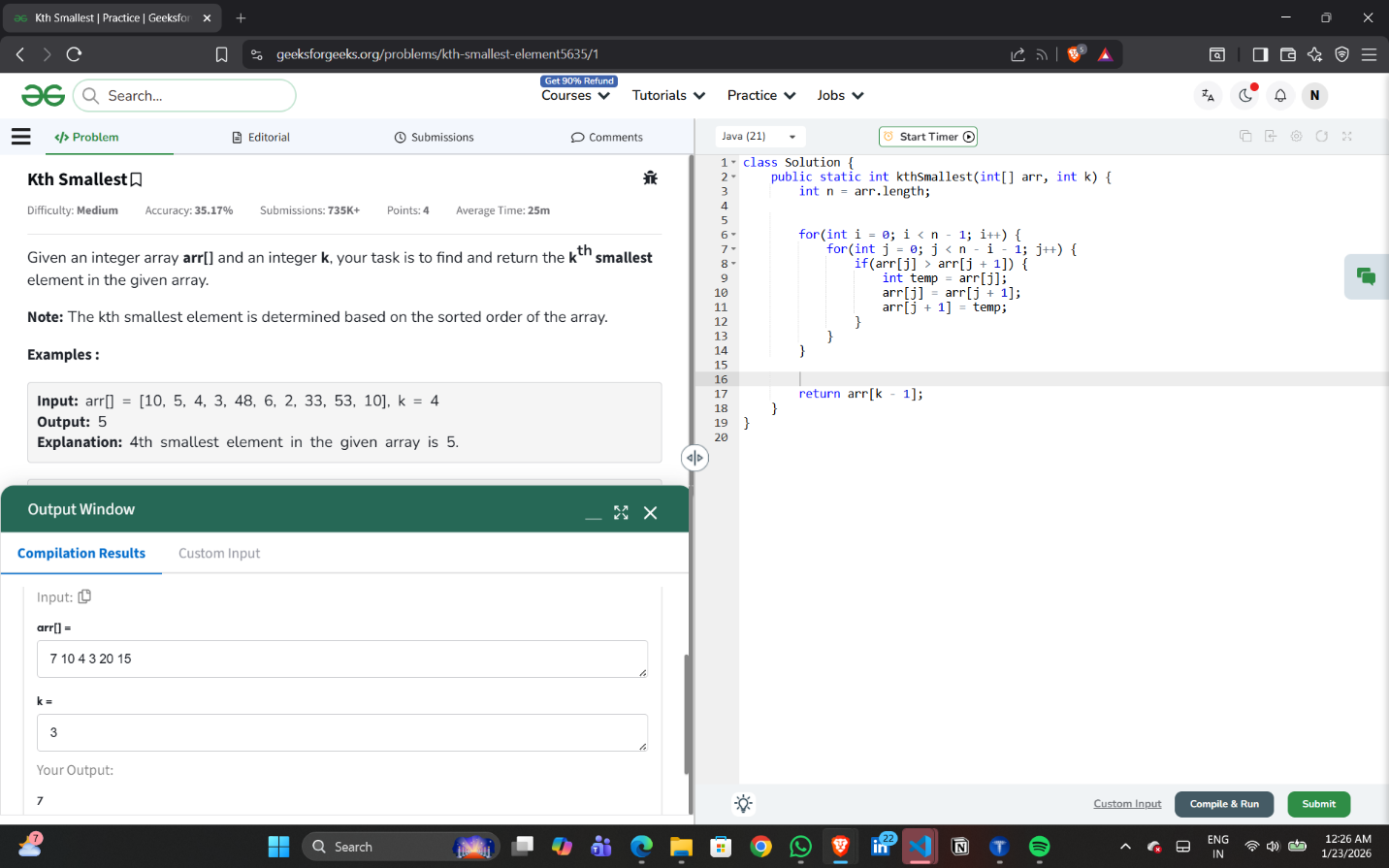
**Output:**



**Experiment 1 – Question 3**  
**Title: Kth Smallest Element in an Array**

**Algorithm:**

1. Sort the array in ascending order using swapping.
2. Return the element at position k-1.

Output:  


**Experiment 1 – Question 5**  
**Title: Largest Element in an Array**

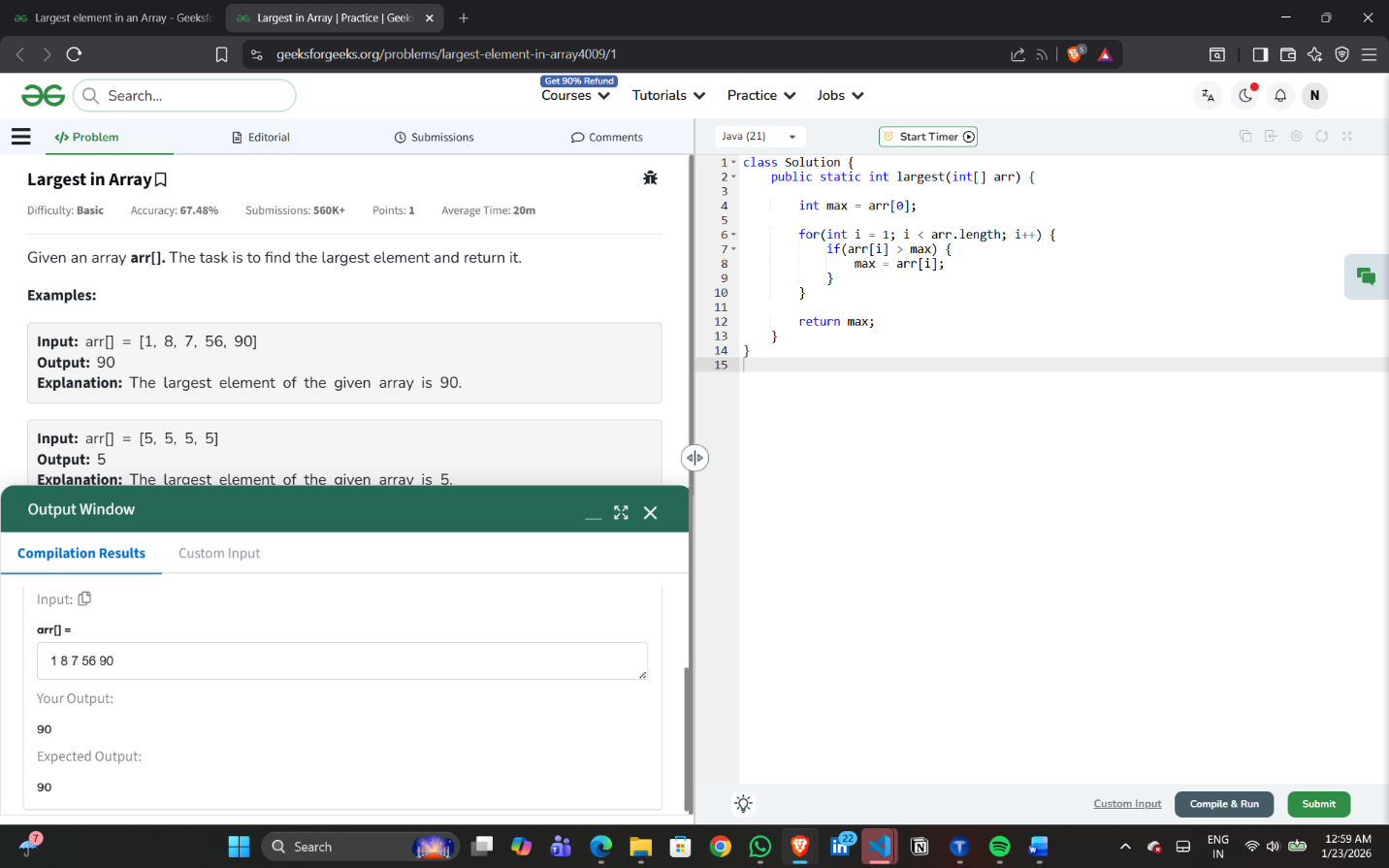
**Algorithm:**

**1. Initialize maximum as first element.**

**2. Traverse the array.**

**3. Update maximum if a larger element is found.**

**4. Return maximum.**

**Output:  
  
**

**Experiment 1 – Question 6**  
**Title:** **Cyclically Rotate an Array by One (Clockwise)**

**Algorithm:**

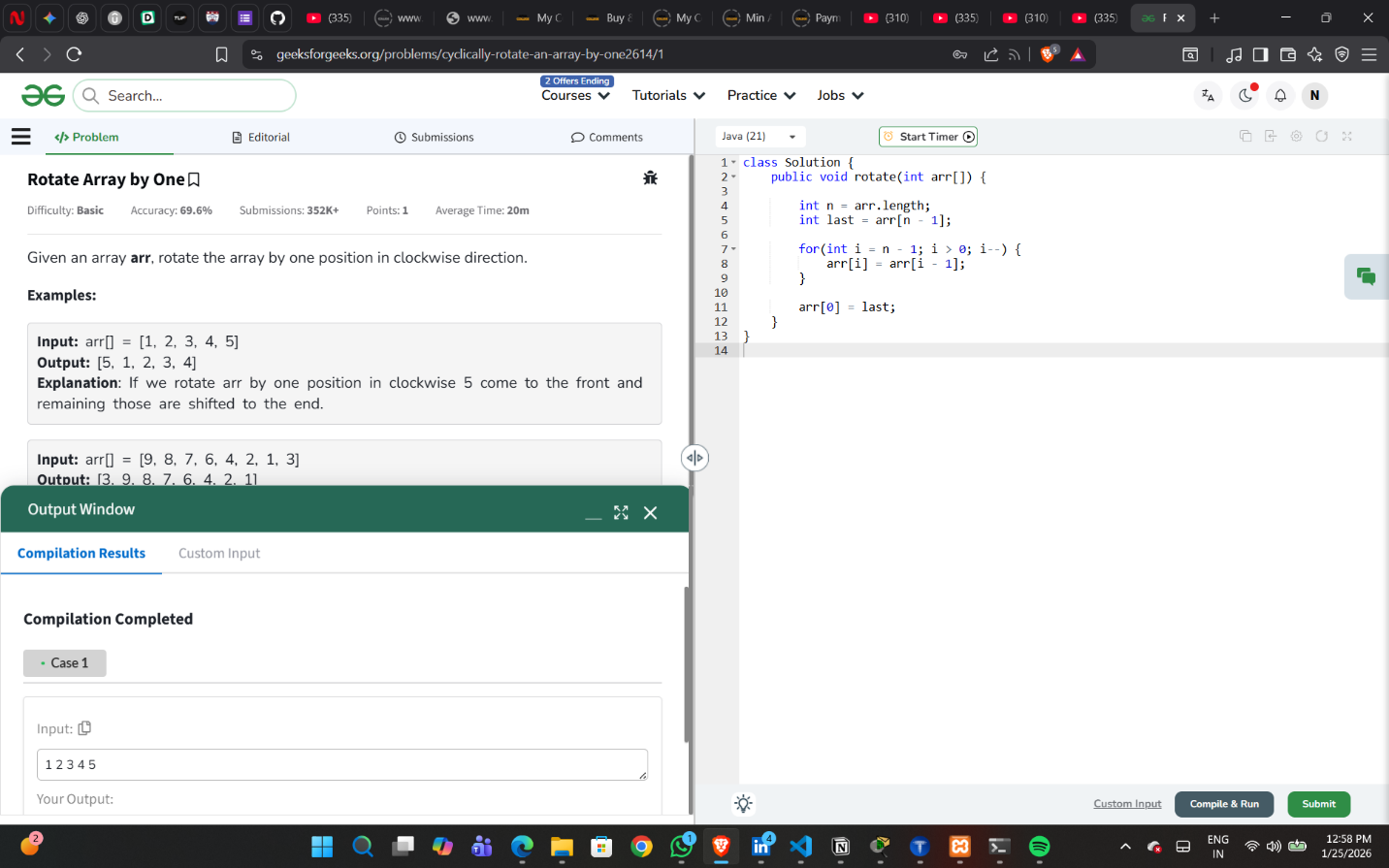
**1. Store the last element of the array in a temporary variable.**

**2. Shift all elements of the array one position to the right.**

**3. Place the stored last element at the first position of the array.**

**4. Display the rotated array.**

**Output:**

****

**Experiment 1 – Question 7**  
**Title**: **Maximum Subarray Sum (Kadane’s Algorithm)**

**Algorithm:**

1. **Initialize maxSum with minimum integer value.2. Shift all elements of the array one position to the right.**
2. **Initialize currentSum as 0.**

**3.** **Traverse the array from first element to last.**

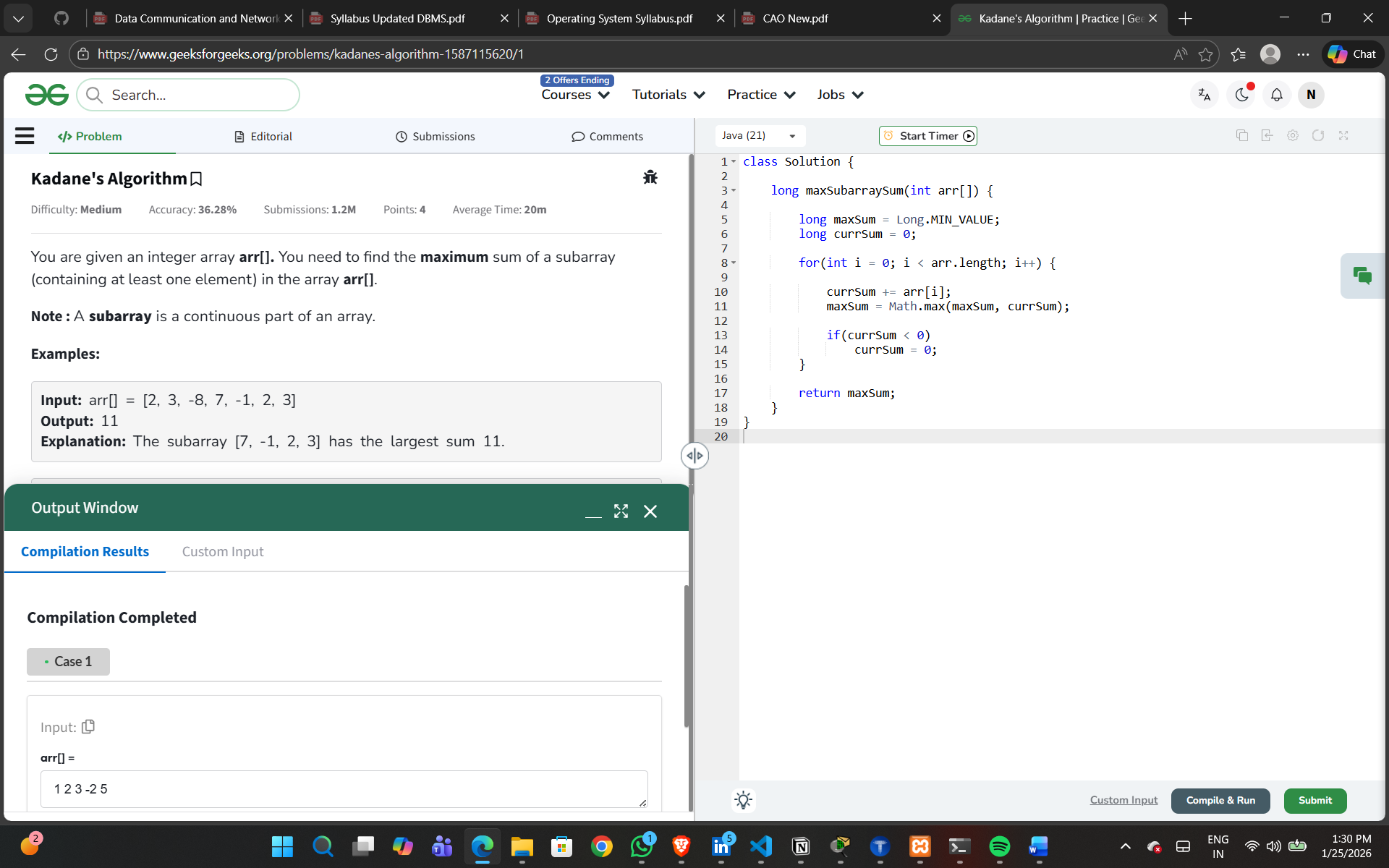
**4. Add each element to currentSum.**

**5.** **Update maxSum if currentSum is greater.**

**6.** **If currentSum becomes negative, reset it to 0.**

**7. After traversal, print maxSum.**

**Output:**



**Experiment 1 – Question 8**  
**Title**: **Search Insert Position**

**Algorithm:**

1. Initialize low = 0, high = n-1.

2.While low <= high:

Find mid.

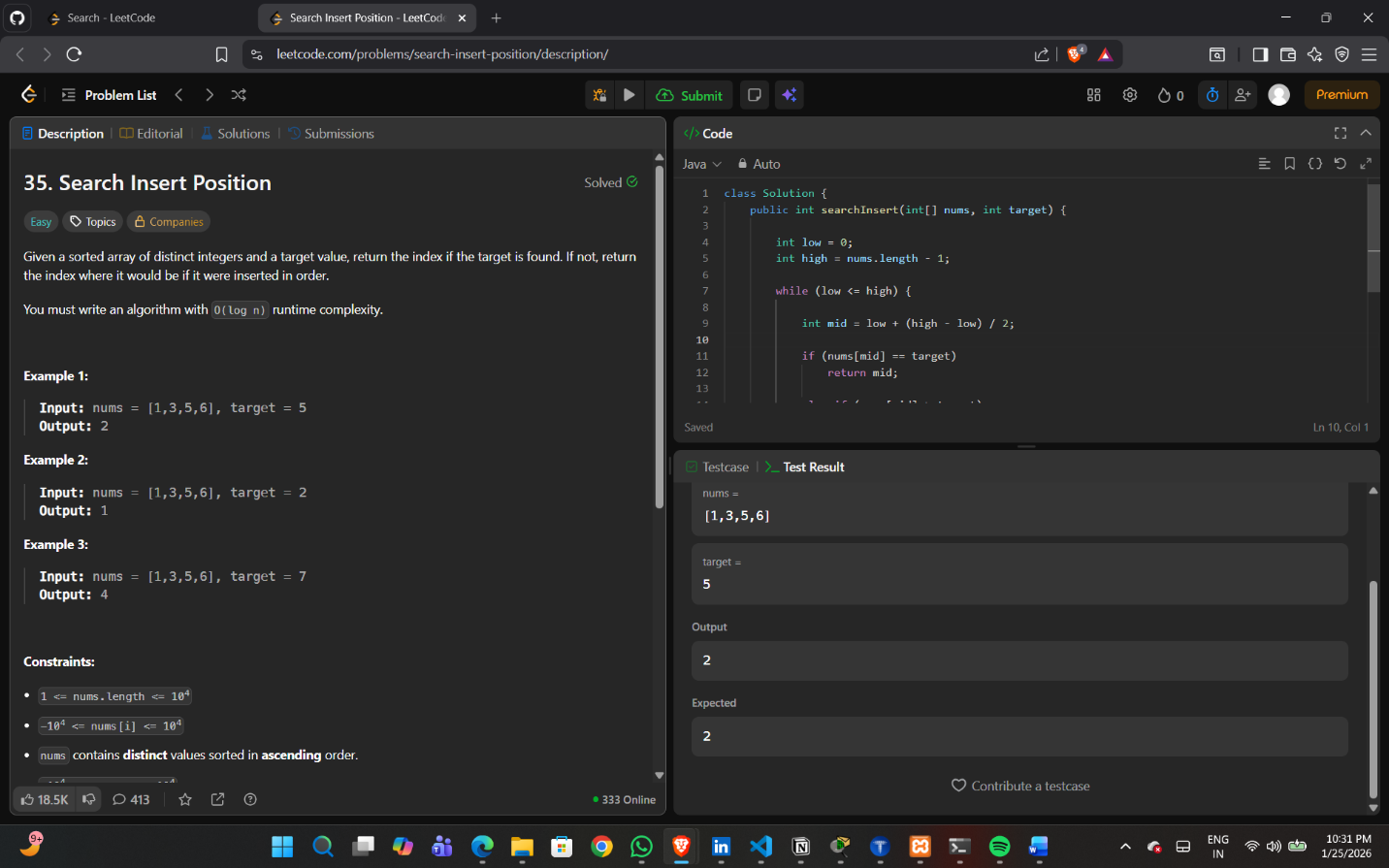
If element found, return mid.

If target is smaller, move high.

Else move low.

1. **Return low**

**Output:**

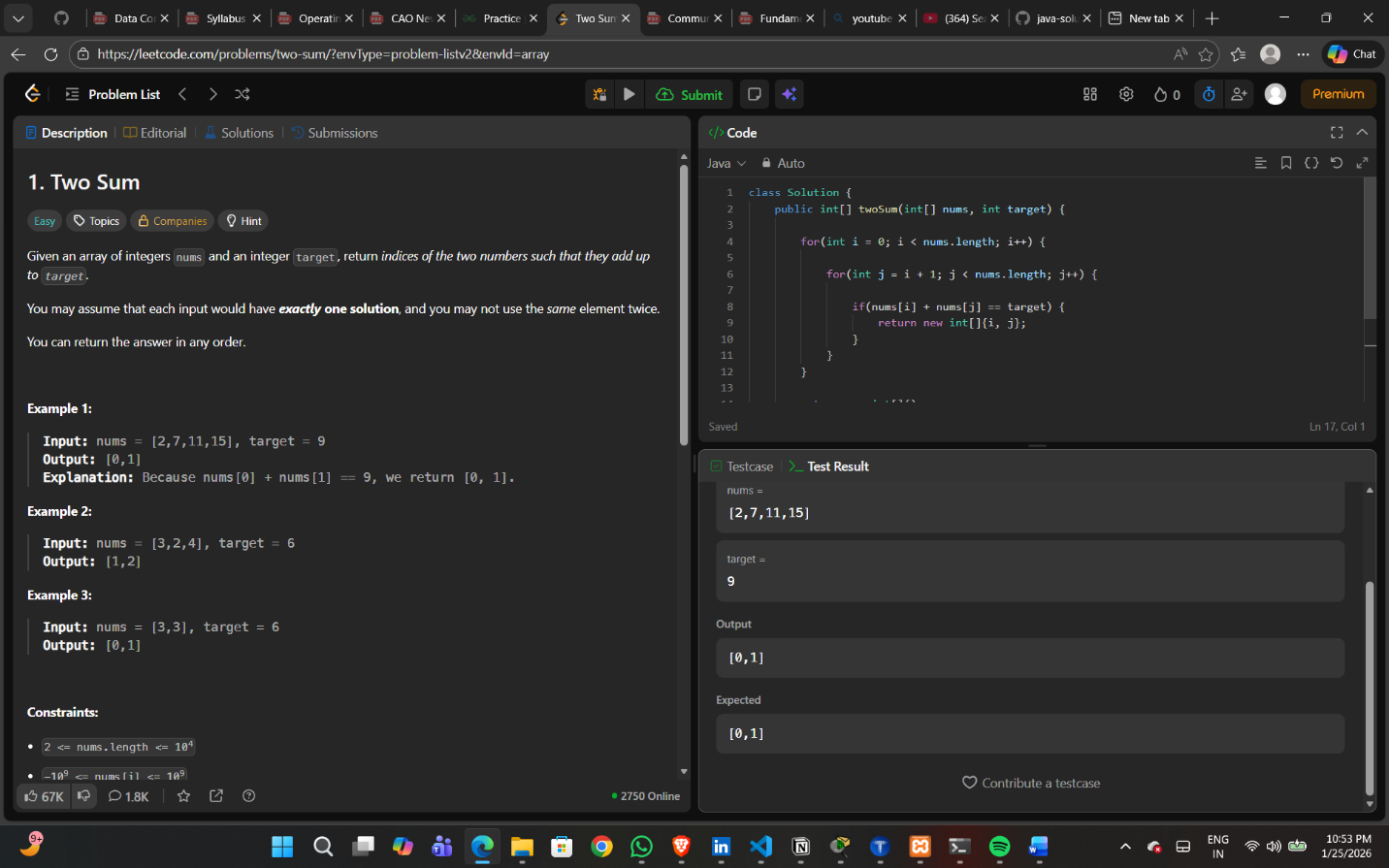
****

**Experiment 1 – Question 9**  
**Title**: Two Sum Problem

**Algorithm:**

1. **Use two nested loops.**
2. **First loop picks first element.**
3. **Second loop picks next elements.**
4. **If sum equals target, print indices and stop.**

**Output:**

****

**Experiment 2 – Question 1**

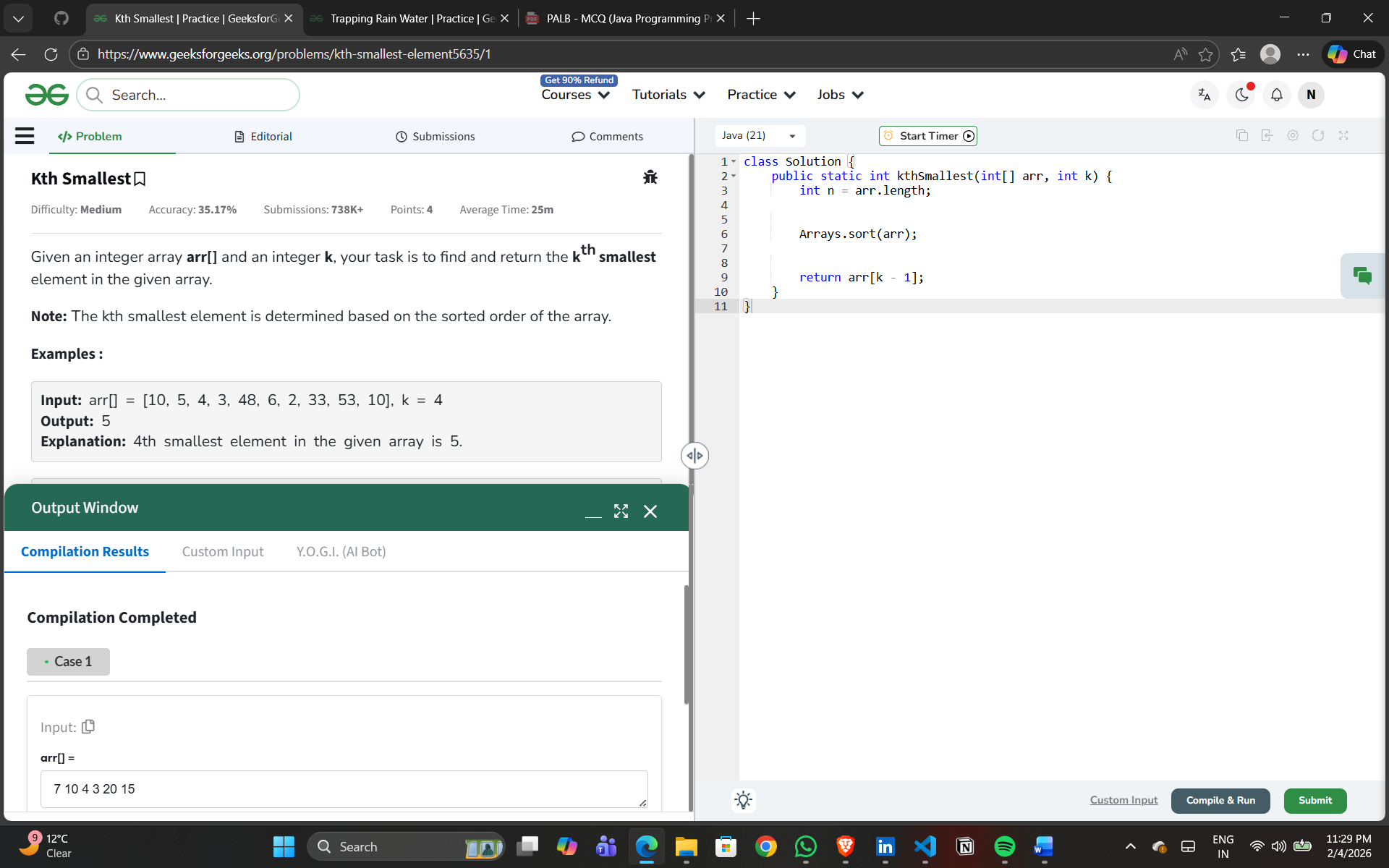
**Title: Kth Smallest Element in Array**

**Algorithm:**

1. Read the array elements and value k.
2. Sort the array in ascending order.
3. Access the element at position (k − 1).
4. Display the kth smallest element.

**Output:**

Kth smallest element is displayed.



**Experiment 2 – Question 2**

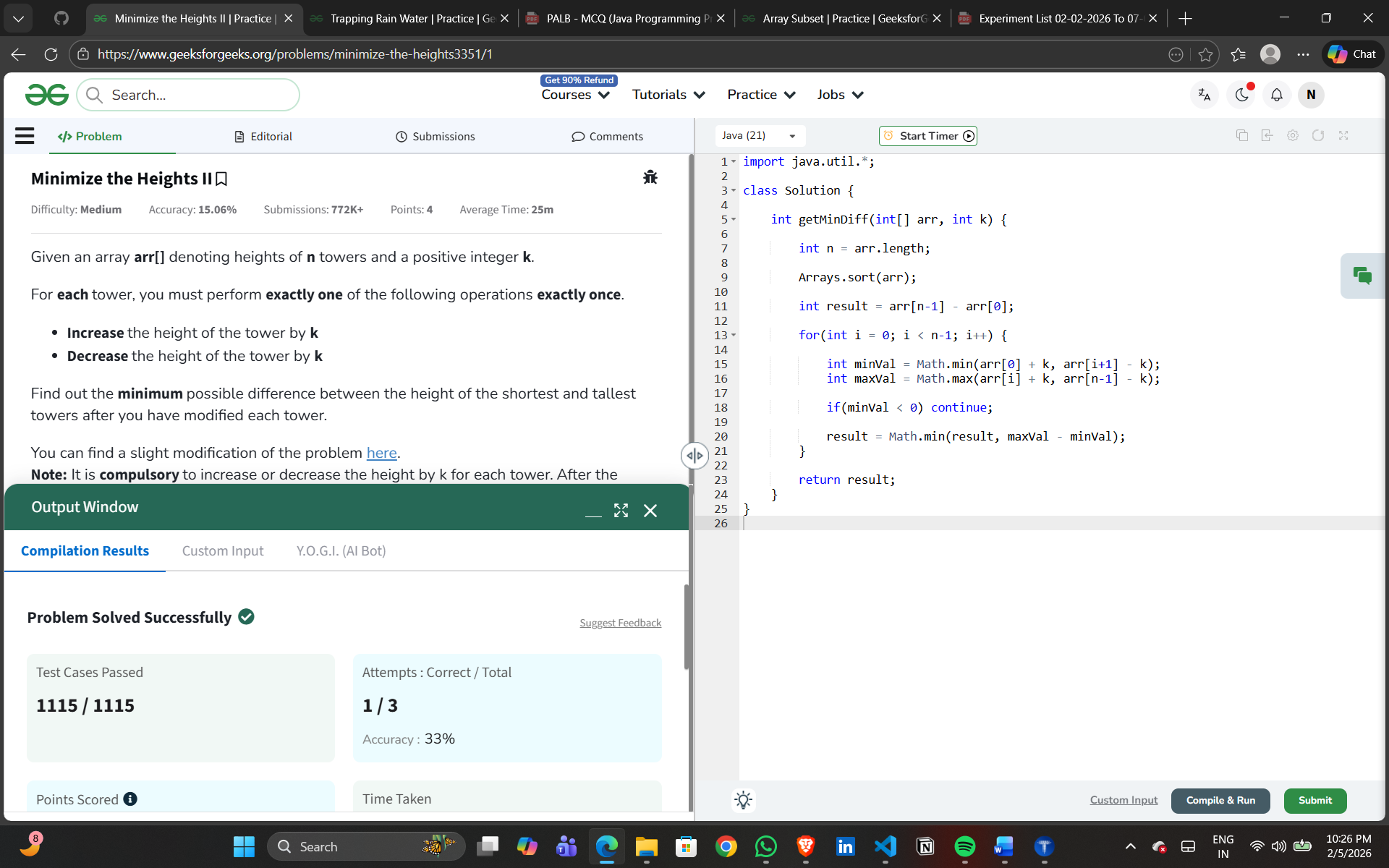
**Title: Minimize the Heights**

**Algorithm:**

1. Input array and value k.
2. Sort the array.
3. Increase the smallest element by k.
4. Decrease the largest element by k.
5. Find new minimum and maximum values.
6. Display the minimum difference.

**Output:**

Minimum possible difference between tower heights is displayed.



**Experiment 2 – Question 3**

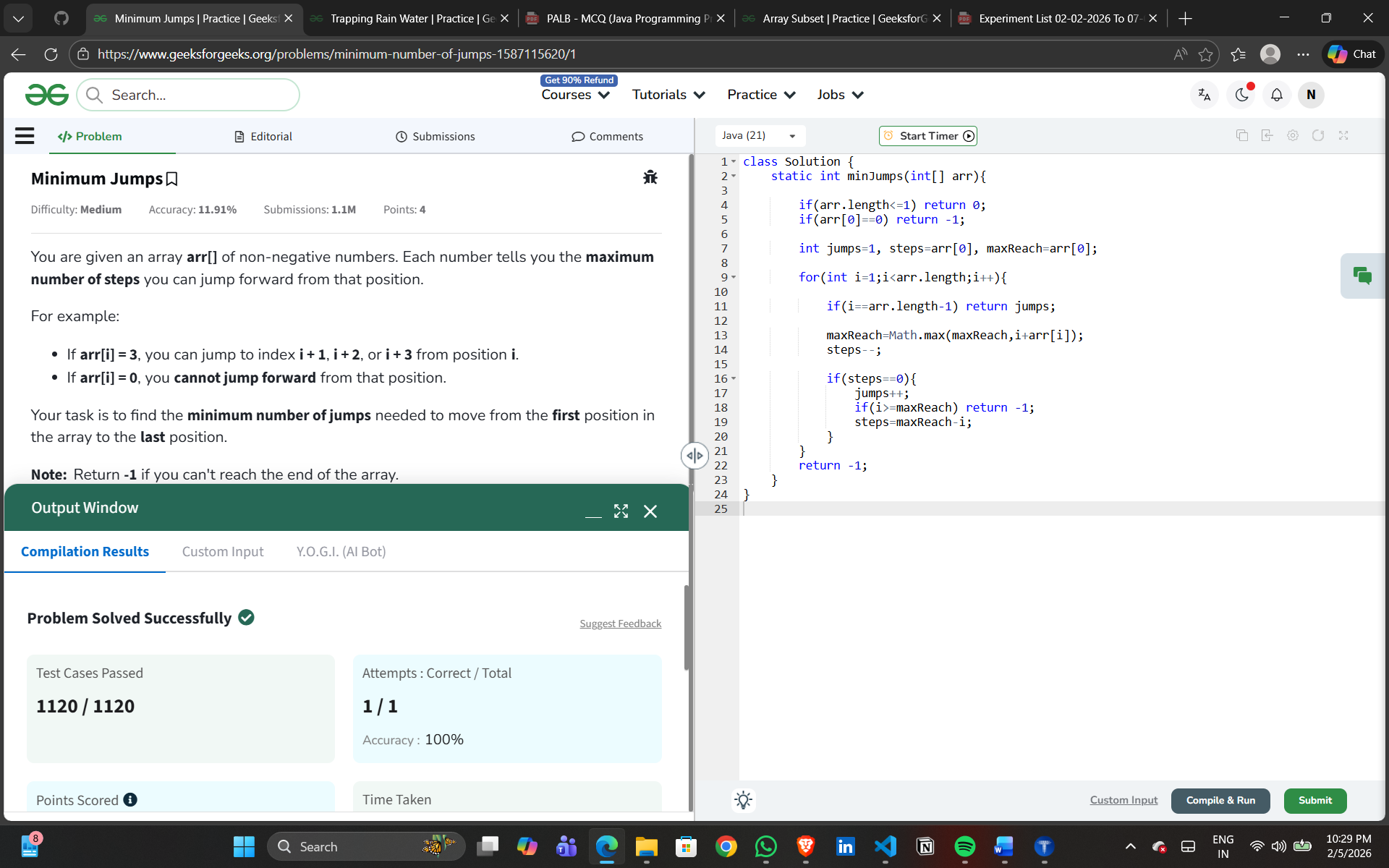
**Title: Minimum Number of Jumps**

**Algorithm:**

1. Initialize maxReach, steps and jumps.
2. Traverse the array.
3. Update maxReach and decrease steps.
4. When steps become zero, increment jump count.
5. Repeat until end of array is reached.
6. Display minimum jumps or −1 if unreachable.

**Output:**

Minimum number of jumps required is displayed.



**Experiment 2 – Question 4**

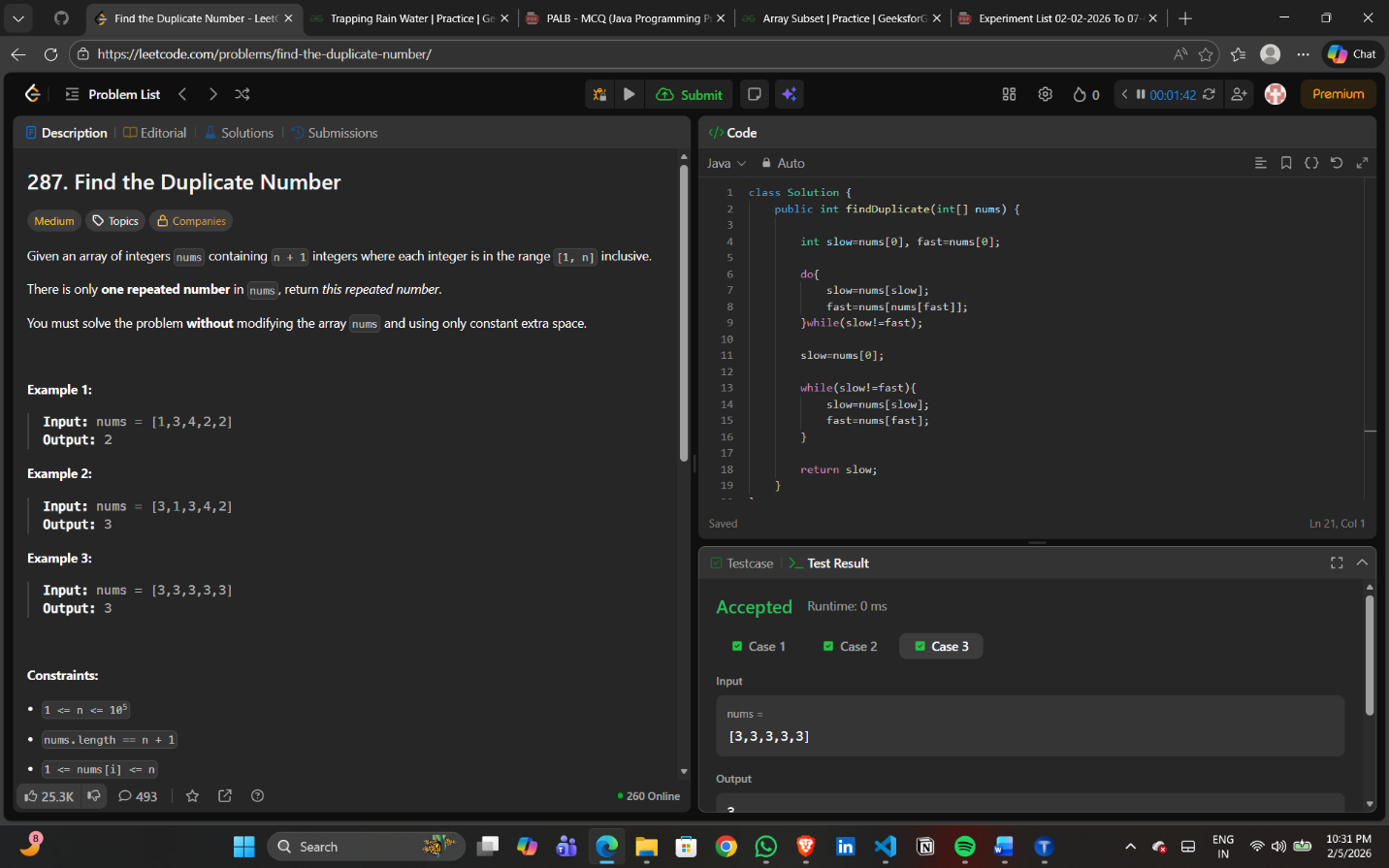
**Title: Find the Duplicate Number**

**Algorithm:**

1. Treat array as linked list.
2. Use slow and fast pointers.
3. Detect cycle using Floyd’s algorithm.
4. Find starting point of cycle.
5. Display duplicate element.

**Output:**

Duplicate number is displayed.



**Experiment 2 – Question 5**

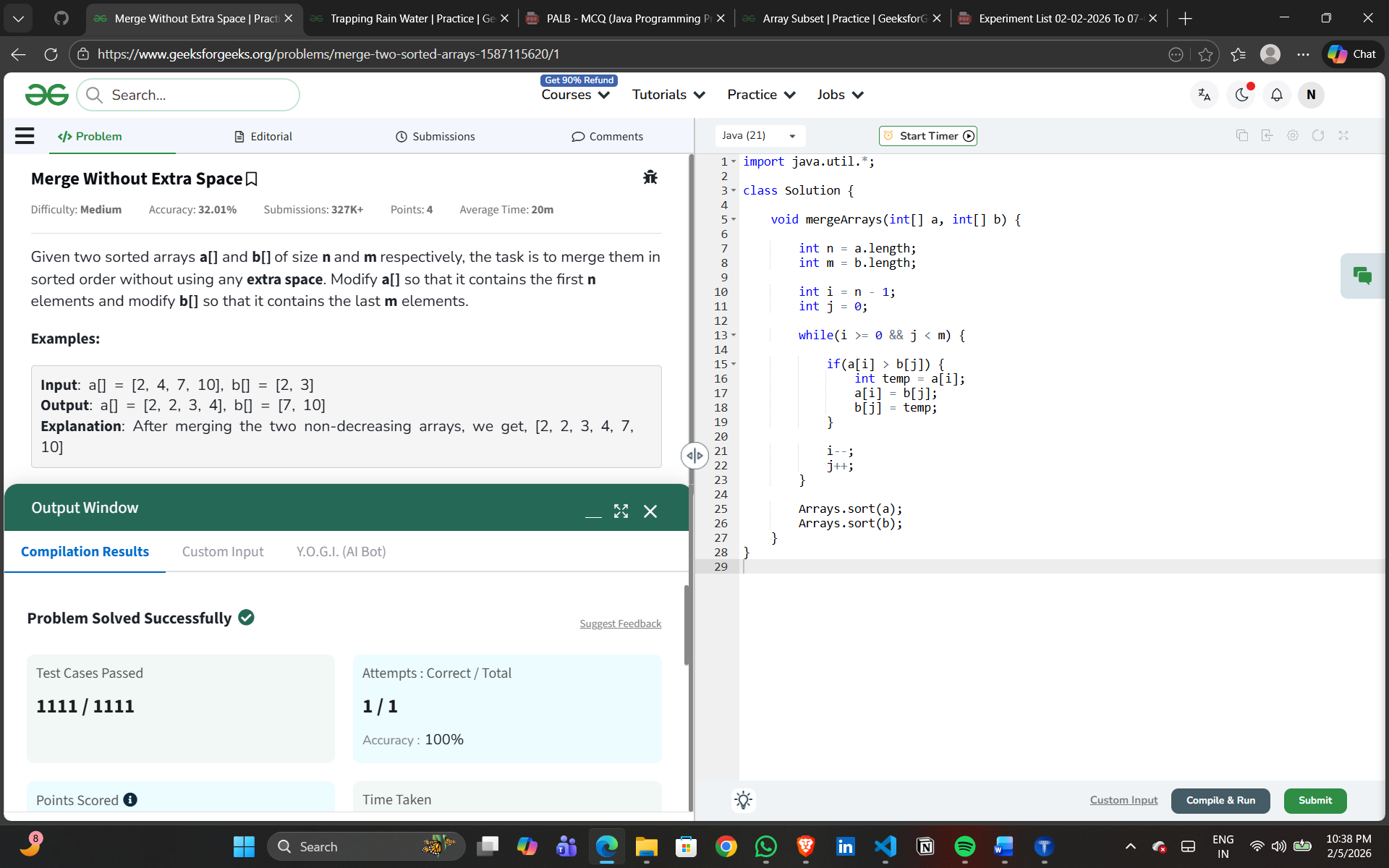
**Title: Merge Two Sorted Arrays Without Extra Space**

**Algorithm:**

1. Compare largest element of first array with smallest of second.
2. Swap if required.
3. Sort both arrays individually.
4. Repeat until arrays are merged.
5. Display both arrays.

**Output:**

Merged sorted arrays are displayed.



**Experiment 2 – Question 6**

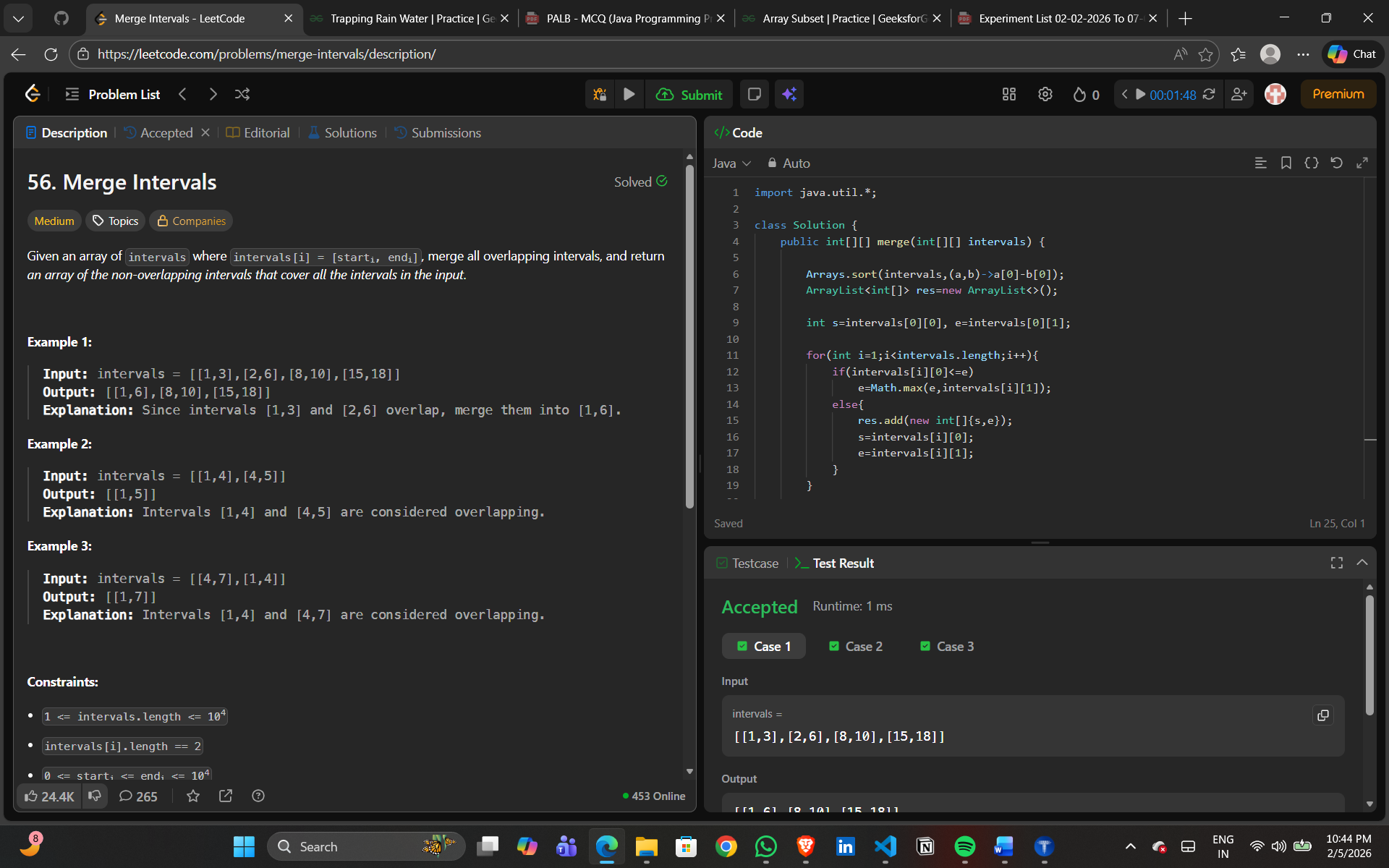
**Title: Merge Intervals**

**Algorithm:**

1. Sort intervals based on starting value.
2. Compare current interval with previous.
3. Merge overlapping intervals.
4. Store non-overlapping intervals.
5. Display merged intervals.

**Output:**

Merged intervals are displayed.



**Experiment 2 – Question 7**

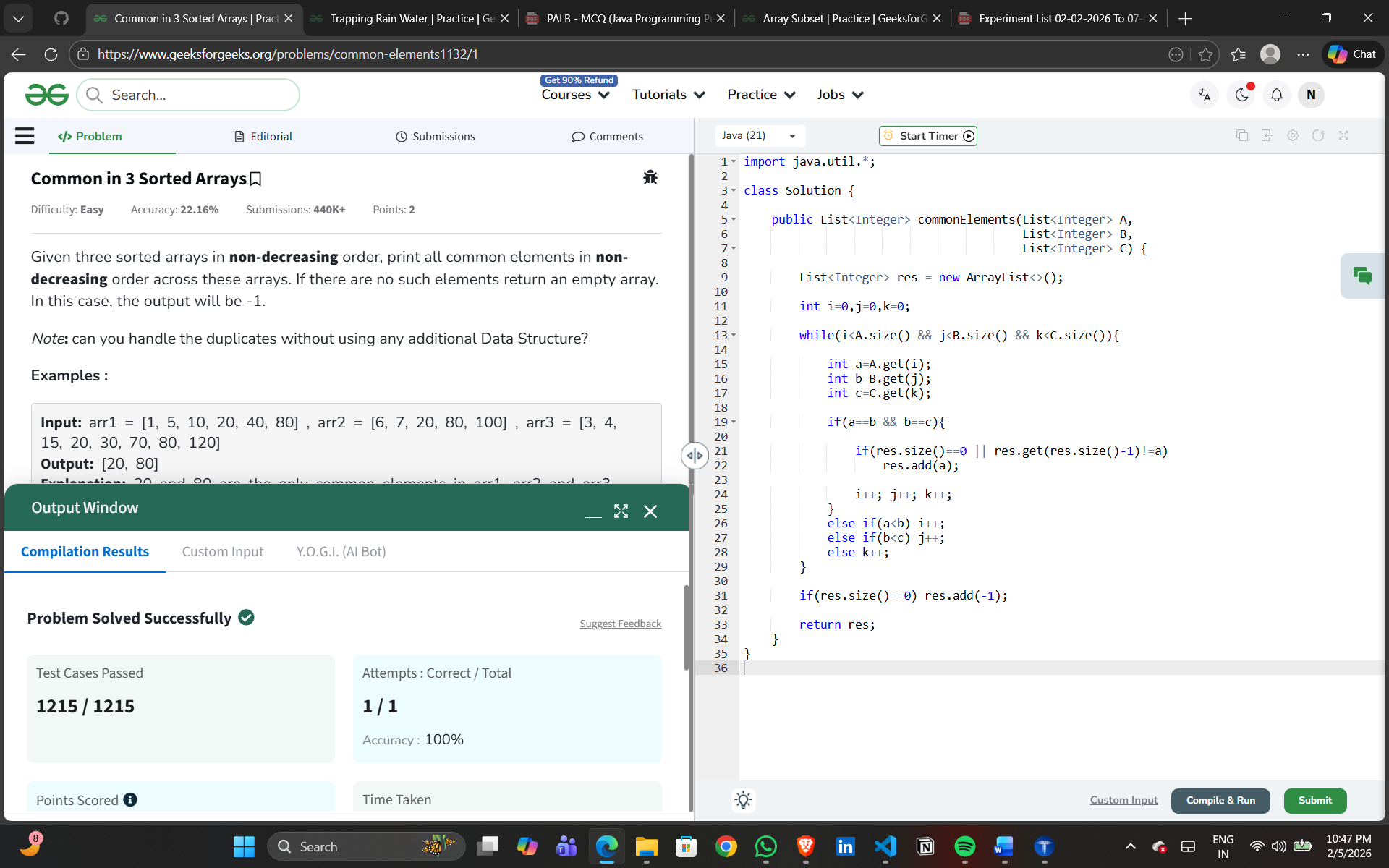
**Title: Common Elements in Three Sorted Arrays**

**Algorithm:**

1. Initialize three pointers i, j, k.
2. Compare elements of all arrays.
3. If equal, store element and move all pointers.
4. If not equal, increment smallest pointer.
5. Repeat until any array ends.

**Output:**

Common elements are displayed or −1 if none.



**Experiment 2 – Question 8**

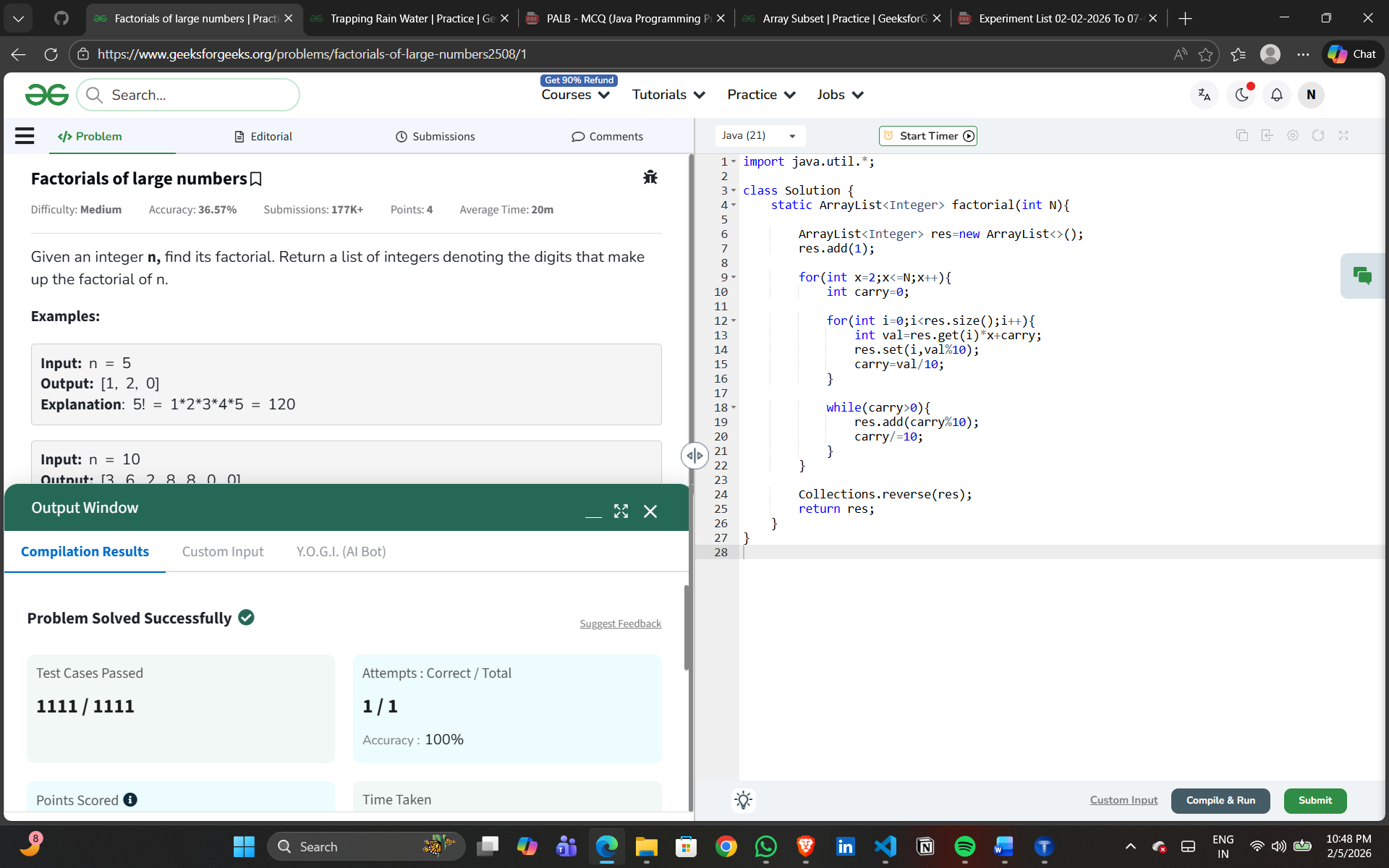
**Title: Factorial of Large Number**

**Algorithm:**

1. Initialize result as 1.
2. Multiply numbers from 2 to n.
3. Store digits separately.
4. Handle carry.
5. Display factorial digits.

**Output:**

Factorial of given number is displayed.



**Experiment 2 – Question 9**

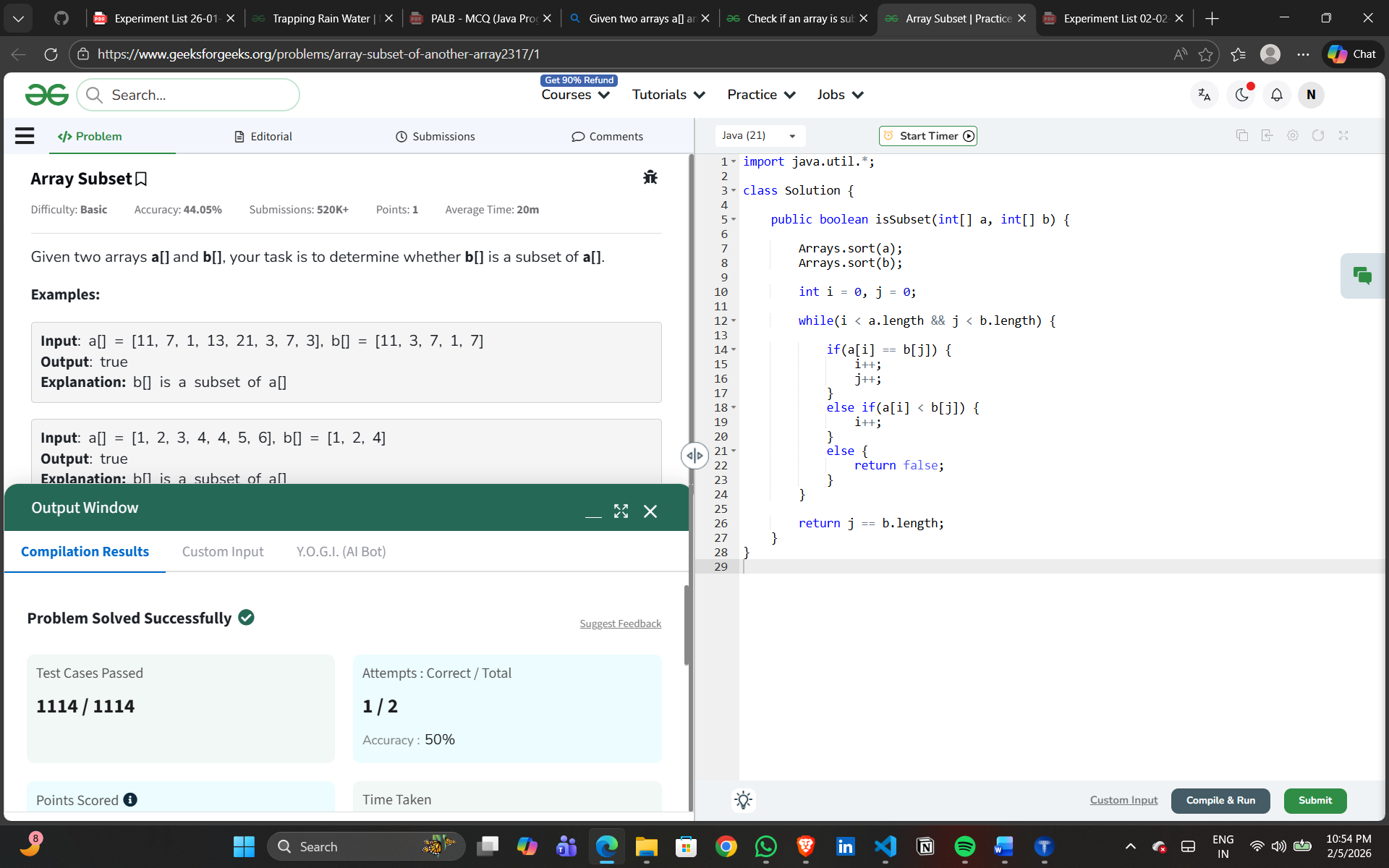
**Title: Array Subset of Another Array**

**Algorithm:**

1. Store elements of first array.
2. Check presence of each element of second array.
3. If any element is missing, return false.
4. Otherwise return true.

**Output:**

Subset result is displayed.



**Experiment 2 – Question 10**

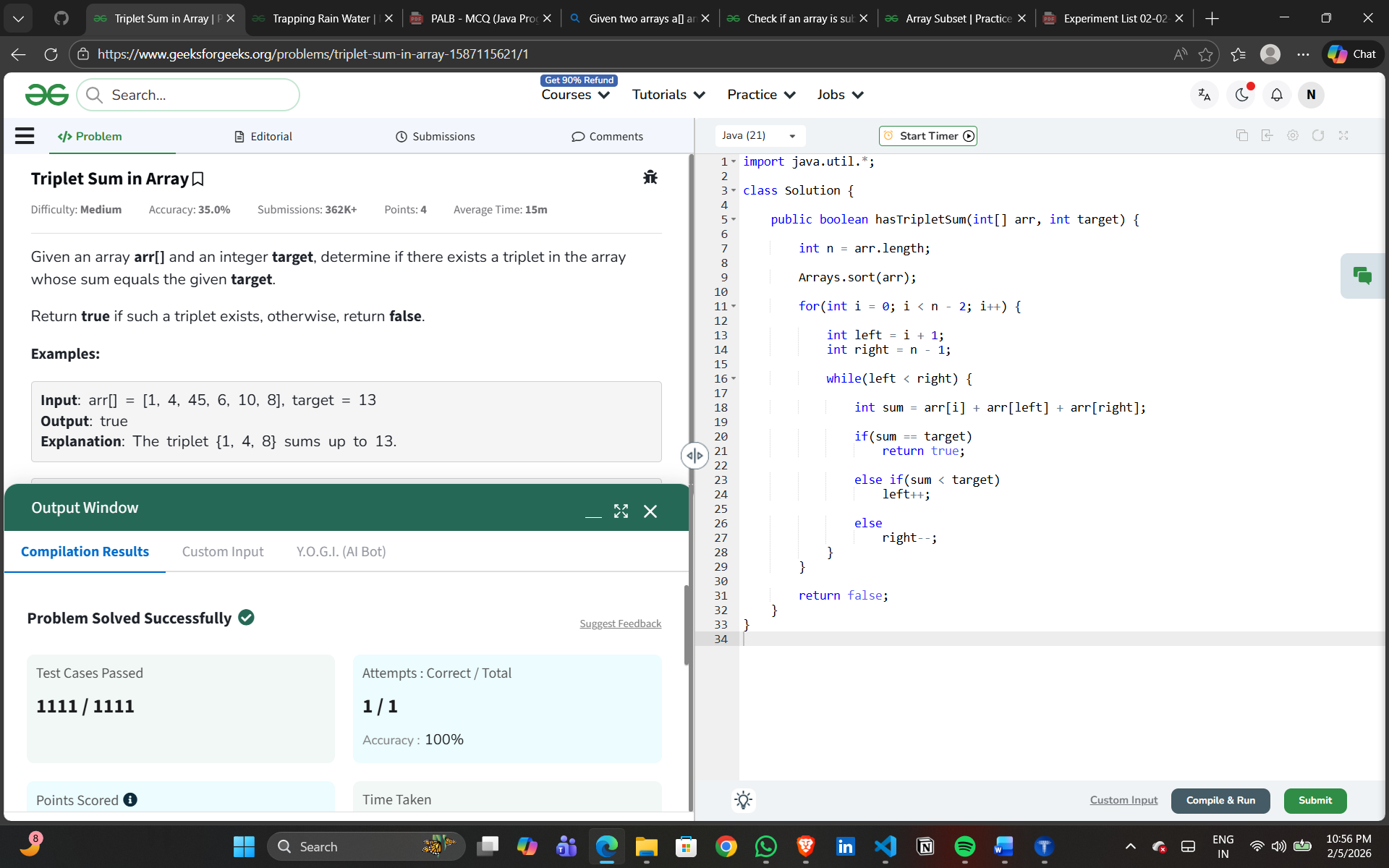
**Title: Triplet Sum in Array**

**Algorithm:**

1. Sort the array.
2. Fix first element.
3. Use two pointers for remaining elements.
4. Check sum with target.
5. Display true if found.

**Output:**

Triplet existence is displayed.



**Experiment 2 – Question 11**

**Title: Trapping Rain Water**

**Algorithm:**

1. Find left maximum for each index.
2. Find right maximum for each index.
3. Calculate water trapped at each position.
4. Add all values.
5. Display total trapped water.

**Output:**

Total trapped rainwater is displayed.

