**// GEEKSFORGEEKS PRACTISE QUESTION //**

**Q1. Count number of elements between two given elements in array**

**class Solution {**

**// Function to get the count of elements between num1 and num2 in array arr.**

**public int getCount(ArrayList<Integer> arr, int num1, int num2) {**

**// Your Code goes here.**

**int leftIndex = arr.indexOf(num1);**

**int rightIndex = arr.lastIndexOf(num2);**

**// Check if the indices are valid and leftIndex is before rightIndex**

**if (leftIndex != -1 && rightIndex != -1 && leftIndex < rightIndex) {**

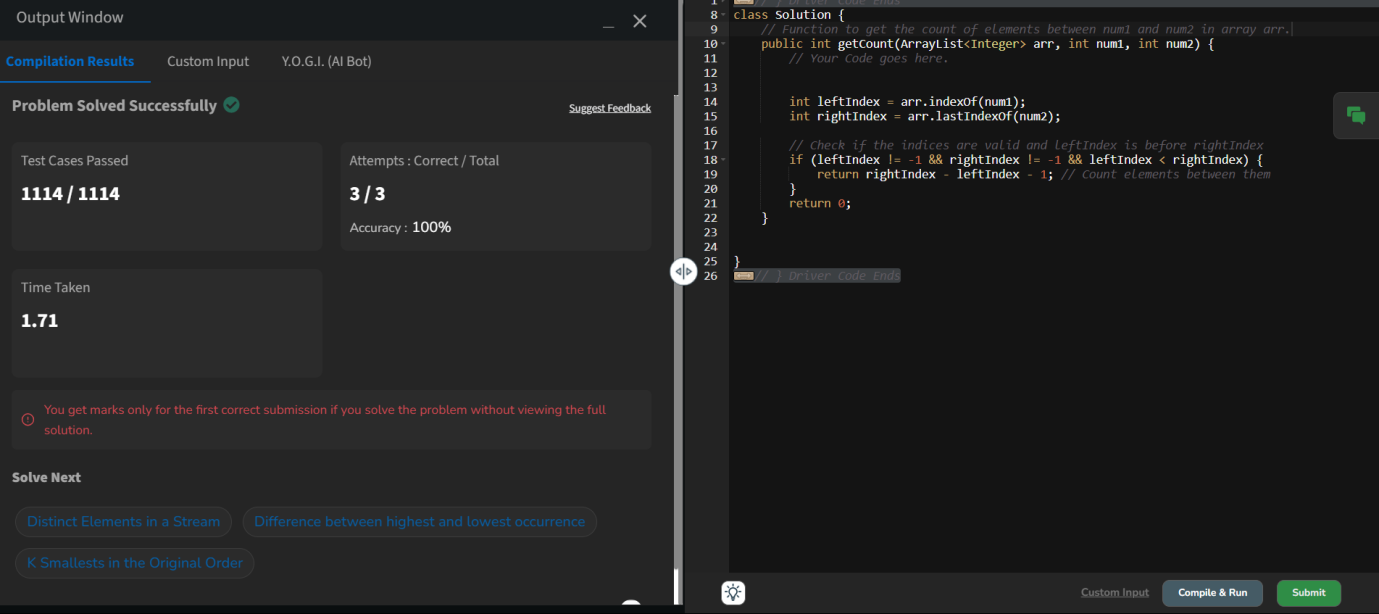
**return rightIndex - leftIndex - 1; // Count elements between them**

**}**

**return 0;**

**}**

}



**Q2.Count Odd Even**

**class Solution {**

**public static in[] countOddEven( int [] arr){**

**int oddCount = 0;**

**int evenCount = 0;**

**for ( int num : arr){**

**if ( num % 2 == 0){**

**evenCount++;**

**}**

**else{**

**oddCount++;**

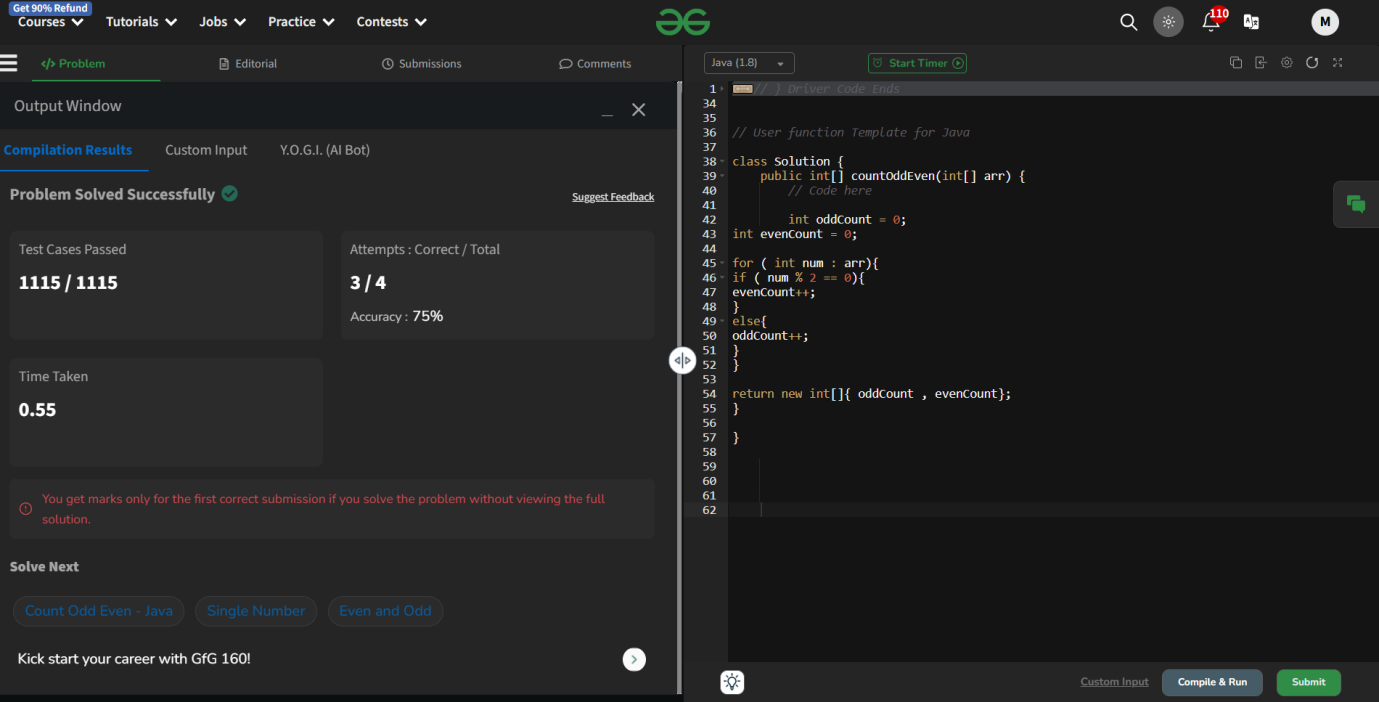
**}**

**}**

**return new int[]{ oddCount , evenCount};**

**}**

**}**

****

Q3. Implement stack using array

**class MyStack {**

**private int[] arr;**

**private int top;**

**// Constructor to initialize stack**

**public MyStack() {**

**arr = new int[1000]; // Initialize array with size 1000**

**top = -1; // Initialize top to -1**

**}**

**// Push operation**

**public void push(int x) {**

**if (top == arr.length - 1) {**

**System.out.println("Stack Overflow"); // Handle overflow if required**

**} else {**

**arr[++top] = x; // Increment top and add element**

**}**

**}**

**// Pop operation**

**public int pop() {**

**if (top == -1) {**

**return -1; // Return -1 if the stack is empty**

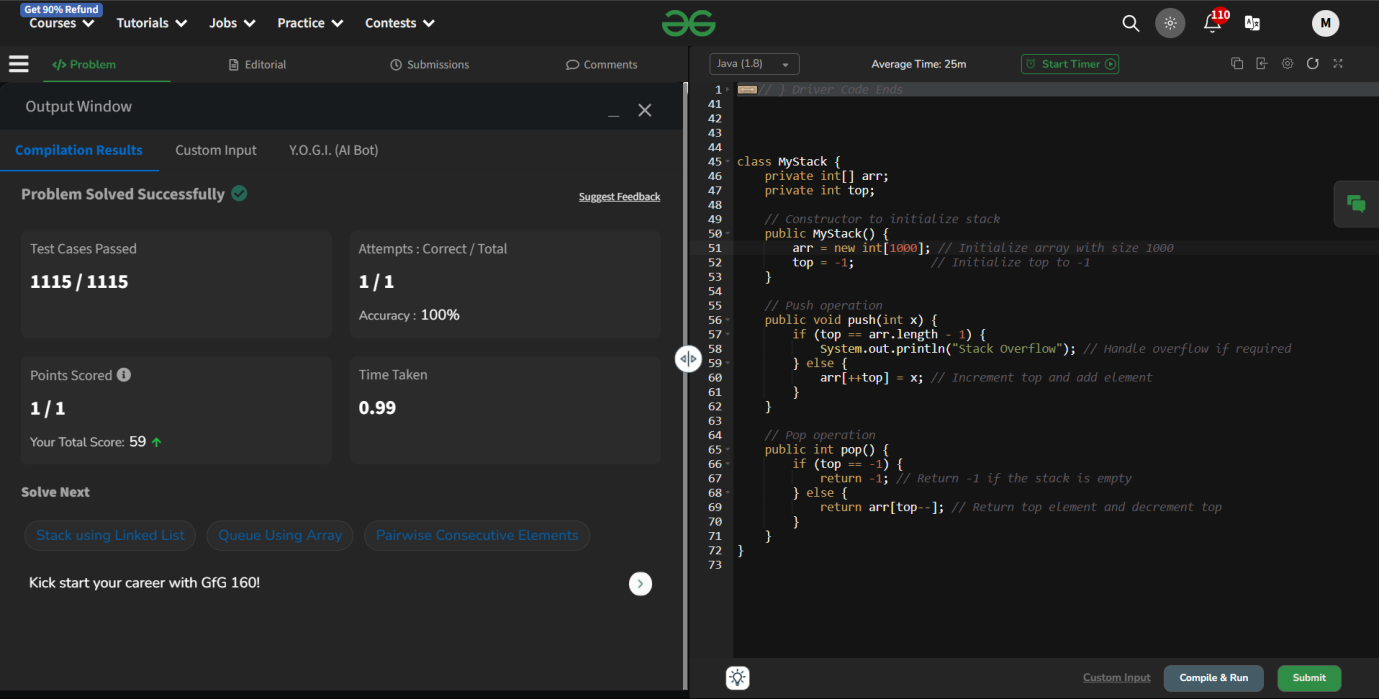
**} else {**

**return arr[top--]; // Return top element and decrement top**

**}**

**}**

**}**

****

Q4. Maximum weight difference

class Solution {

public long MaxWeightDiff(int[] arr, int k) {

int n = arr.length;

// Sort the array

Arrays.sort(arr);

// Case 1: Sum of k smallest elements

int sumKSmallest = 0;

for (int i = 0; i < k; i++) {

sumKSmallest += arr[i];

}

// Case 2: Sum of k largest elements

int sumKLargest = 0;

for (int i = n - k; i < n; i++) {

sumKLargest += arr[i];

}

// Total sum of array

int totalSum = 0;

for (int num : arr) {

totalSum += num;

}

// Calculate the differences

int diff1 = Math.abs(sumKSmallest - (totalSum - sumKSmallest));

int diff2 = Math.abs(sumKLargest - (totalSum - sumKLargest));

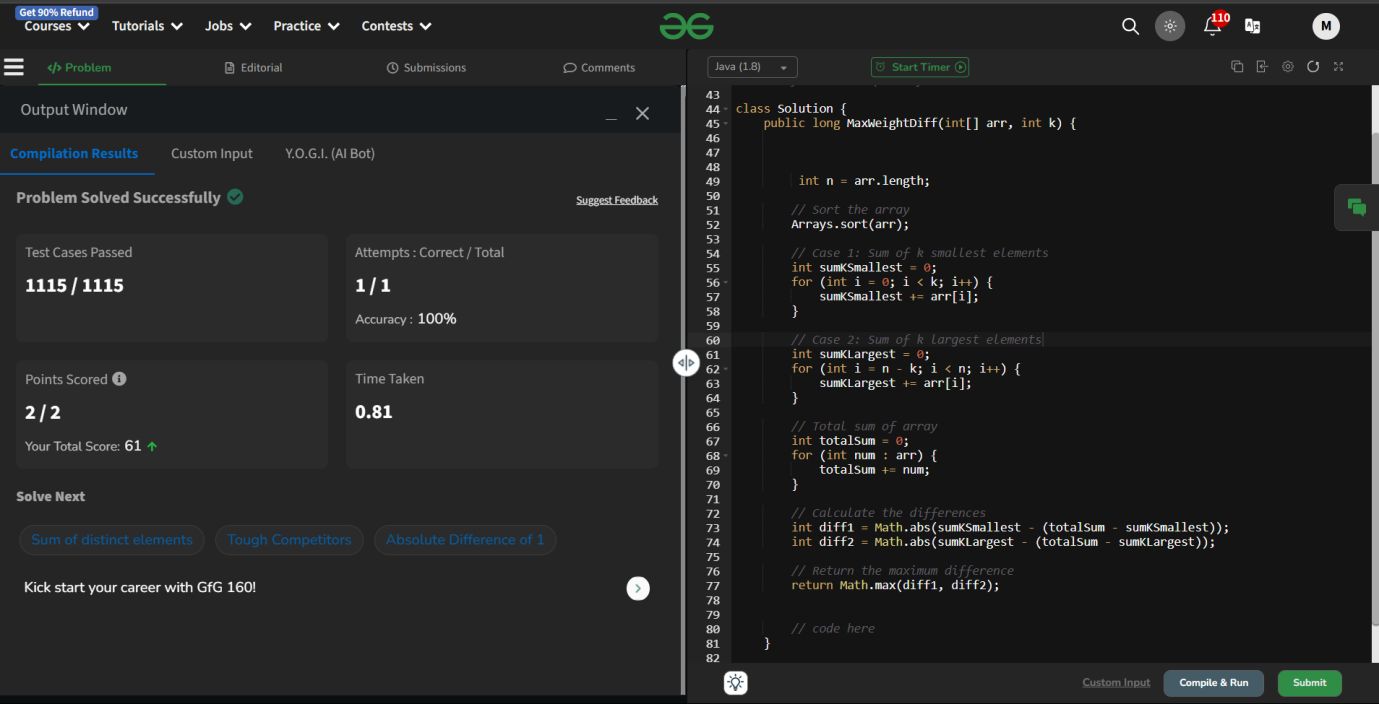
// Return the maximum difference

return Math.max(diff1, diff2);

// code here

}

}



**Q5. Missing number in shuffled array**

**class Solution {**

**public int findMissing(int[] arr1, int[] arr2) {**

**// code here**

**int sum1 = 0;**

**for ( int num : arr1){**

**sum1 += num;**

**}**

**int sum2 = 0;**

**for ( int num : arr2){**

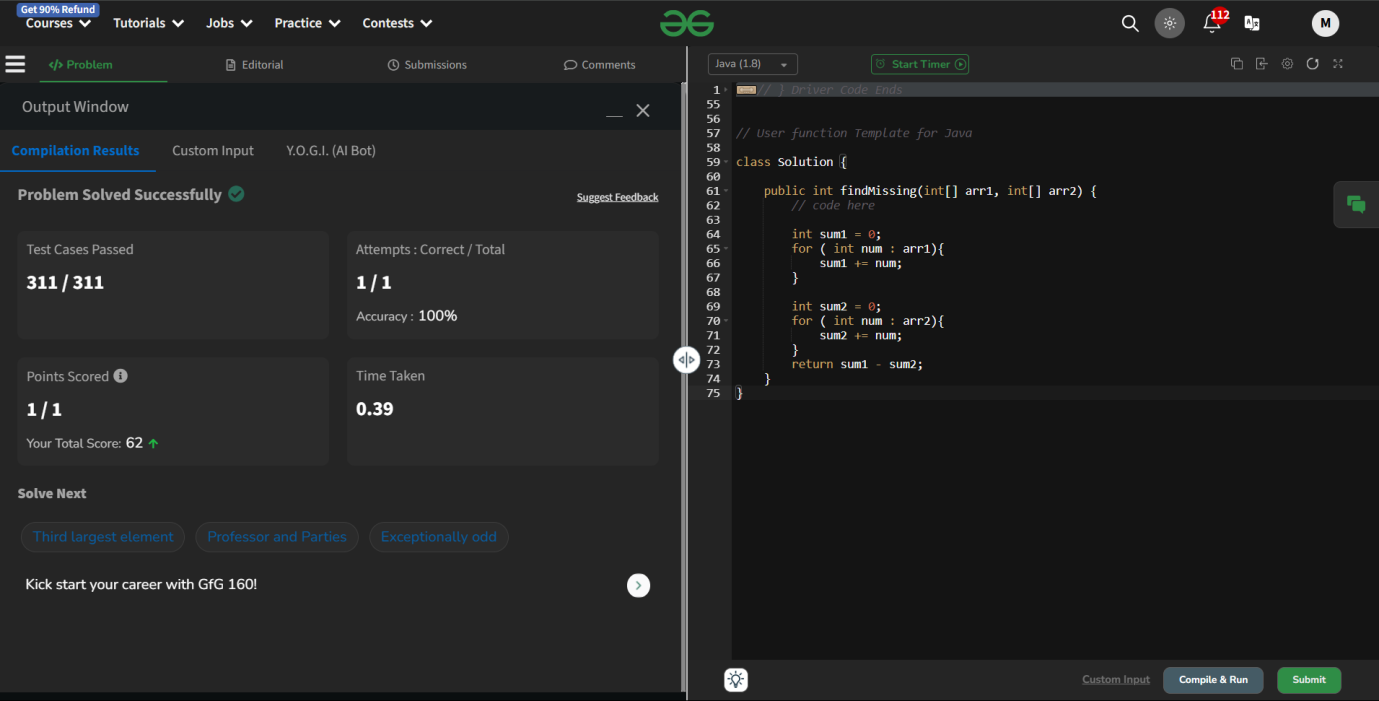
**sum2 += num;**

**}**

**return sum1 - sum2;**

**}**

**}**

****

**Q6. Perfect Array**

**class Solution {**

**public boolean isPerfect(int[] arr) {**

**// code here**

**int n = arr.length;**

**if (n < 2) return true; // An array with less than 2 elements is always perfect.**

**int i = 0;**

**// Check for strictly increasing part**

**while (i < n - 1 && arr[i] < arr[i + 1]) {**

**i++;**

**}**

**// Check for constant part**

**while (i < n - 1 && arr[i] == arr[i + 1]) {**

**i++;**

**}**

**// Check for strictly decreasing part**

**while (i < n - 1 && arr[i] > arr[i + 1]) {**

**i++;**

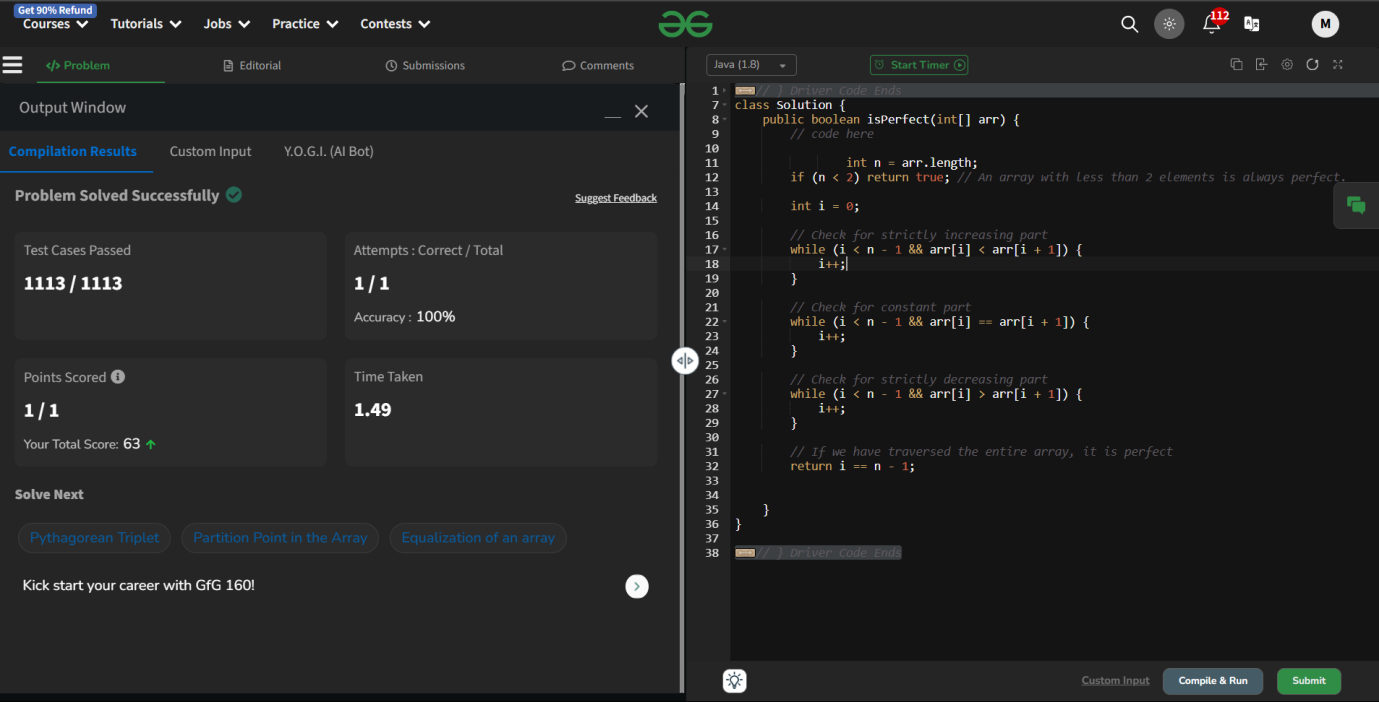
**}**

**// If we have traversed the entire array, it is perfect**

**return i == n - 1;**

**}**

**}**

****

**Q7. Find unique element**

**class Solution {**

**public int findUnique(int k, int[] arr) {**

**// code here**

**int [] bitCount = new int[32];**

**for ( int num : arr){**

**for( int i = 0; i < 32 ; i++){**

**bitCount[i] += (num >> i) & 1;**

**}**

**}**

**int result = 0;**

**for ( int i = 0; i < 32 ; i++){**

**if ( bitCount[i] % k != 0){**

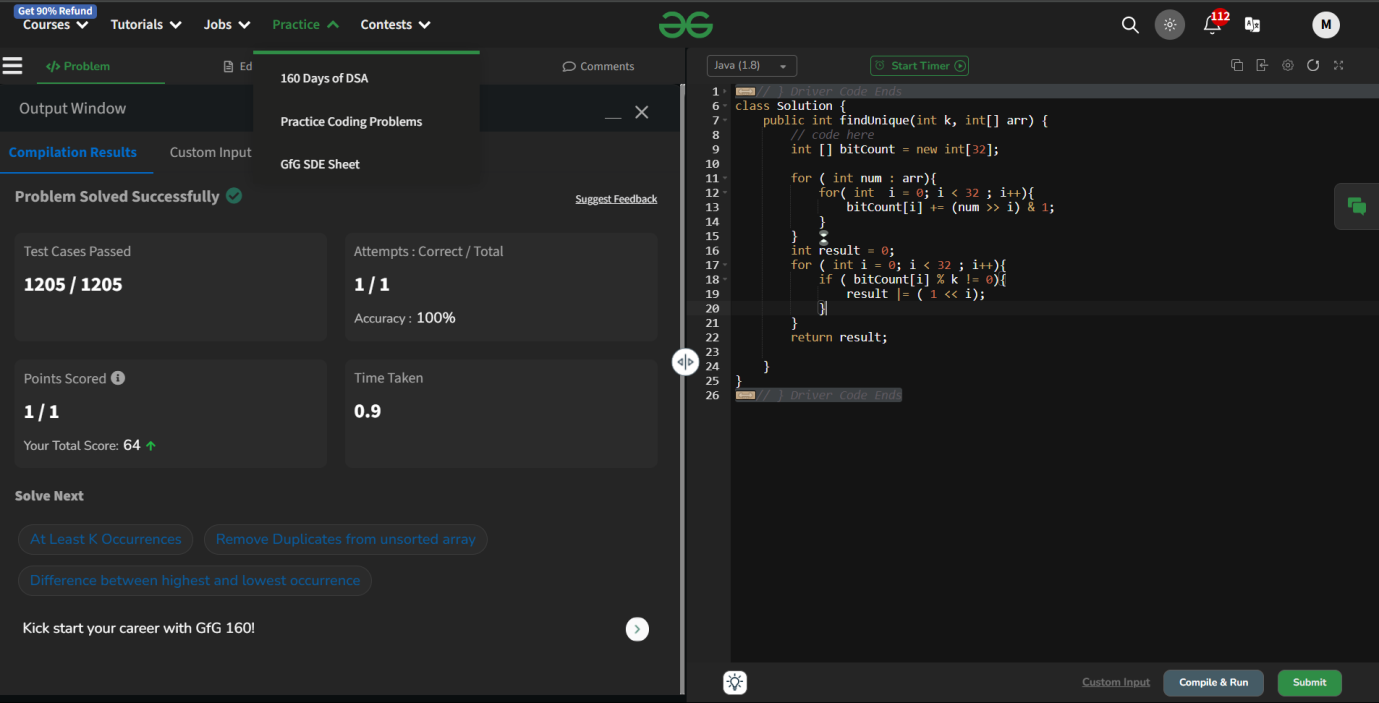
**result |= ( 1 << i);**

**}**

**}**

**return result;**

**}**

**}**

**Q8. Largest Element in Array**

**class Solution {**

**public static int largest(int[] arr) {**

**int max = 0;**

**for ( int i = 0; i<arr.length; i++){**

**if ( arr[i] > max){**

**max = arr[i];**

**}**

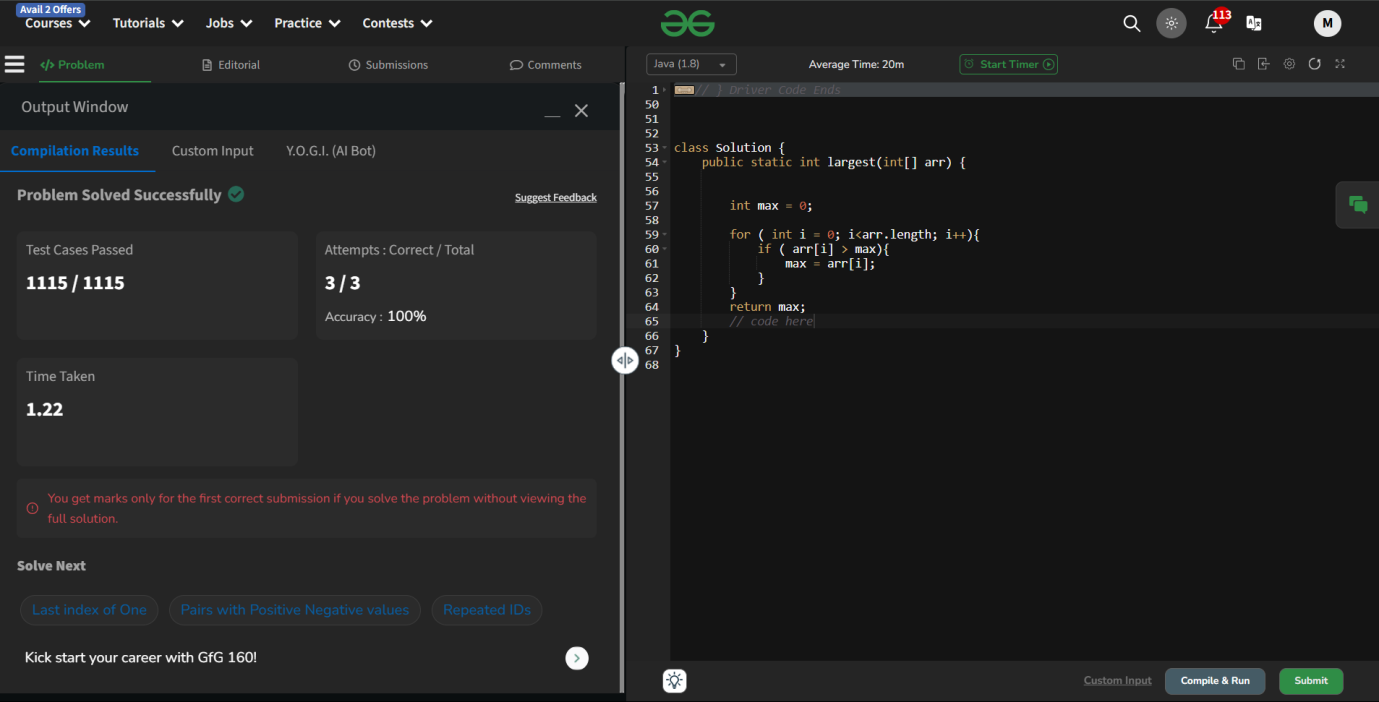
**}**

**return max;**

**// code here**

**}**

**}**

****

**Q9. Check Equal Arrays**

**class Solution {**

**public static boolean checkEqual(int[] a, int[] b) {**

**// Your code here**

**if ( a.length != b.length){**

**return false;**

**}**

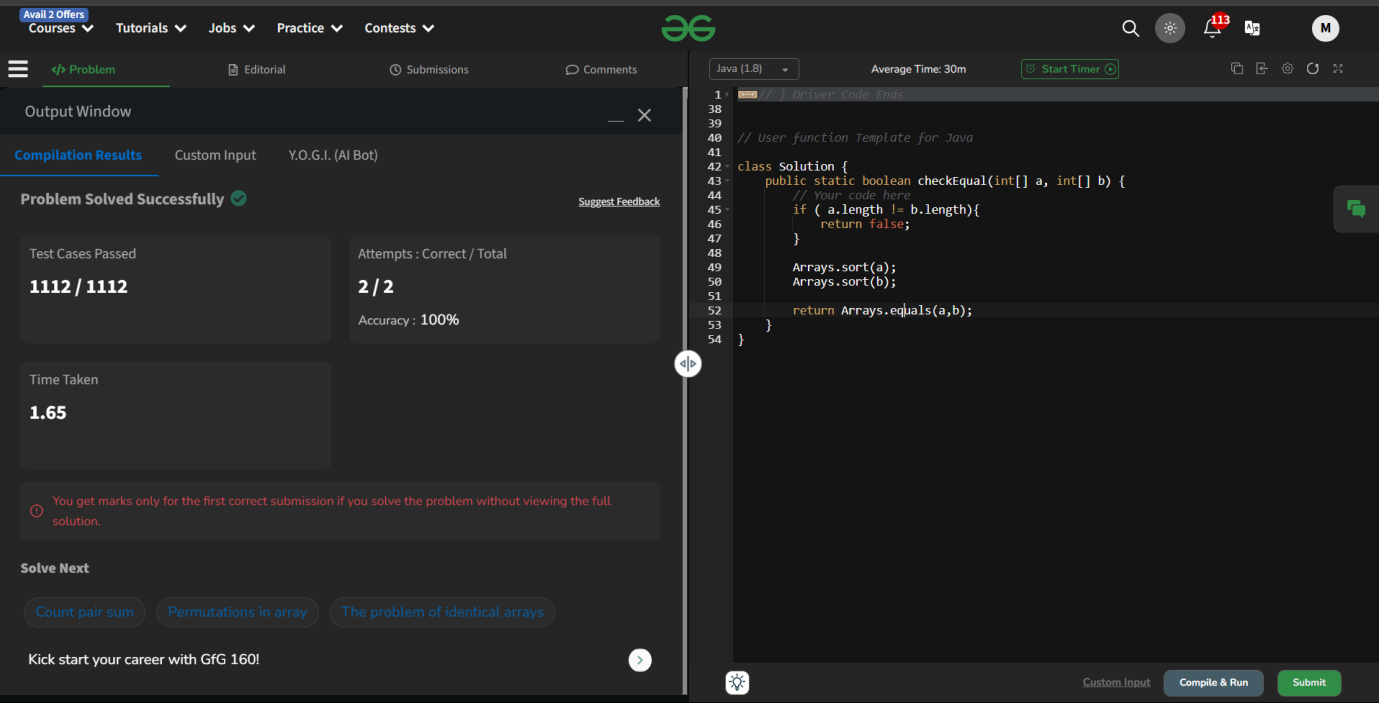
**Arrays.sort(a);**

**Arrays.sort(b);**

**return Arrays.equals(a,b);**

**}**

**}**

****

**Q10. Remove Duplicates from unsorted array**

**class Solution {**

**static ArrayList<Integer> removeDuplicate(int arr[]) {**

**// code here**

**LinkedHashSet<Integer> set = new LinkedHashSet<>();**

**for (int num : arr) {**

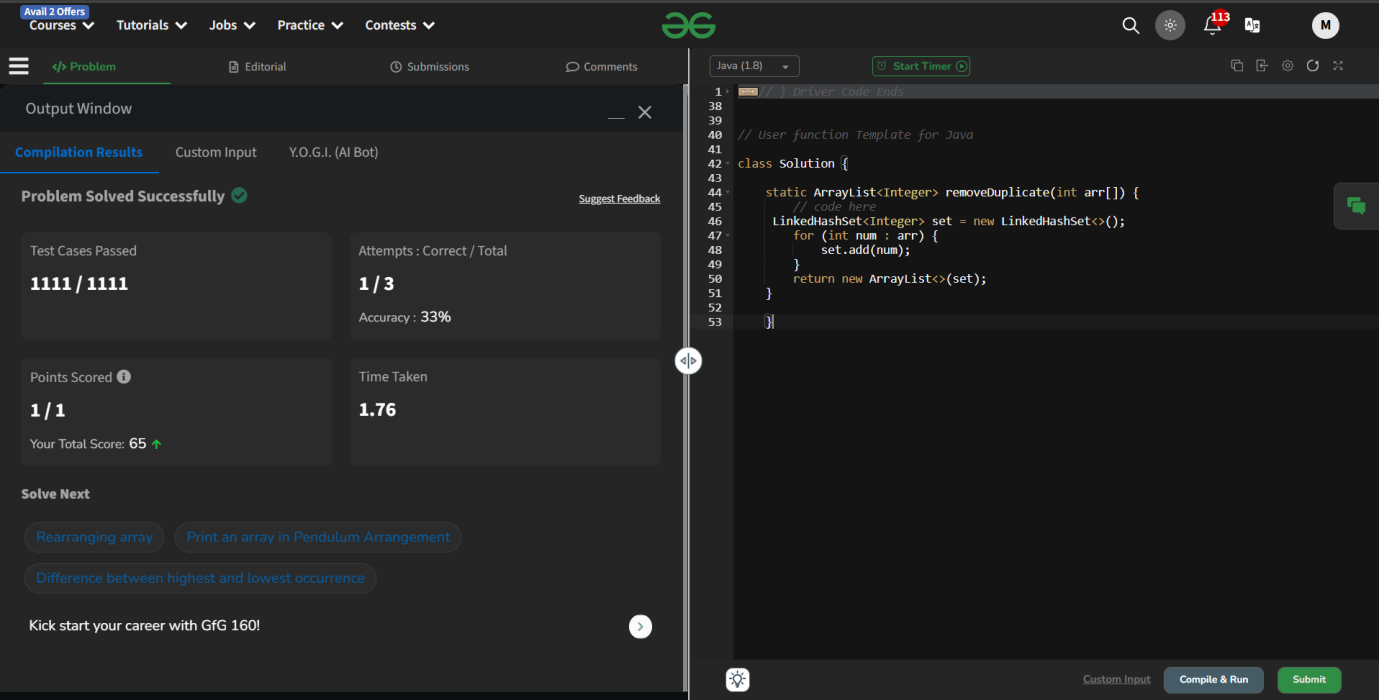
**set.add(num);**

**}**

**return new ArrayList<>(set);**

**}**

**}**

****

**Q11. Positive and negative elements**

**class Solution {**

**ArrayList<Integer> arranged(ArrayList<Integer> arr) {**

**// code here**

**// Separate the positive and negative numbers**

**ArrayList<Integer> positive = new ArrayList<>();**

**ArrayList<Integer> negative = new ArrayList<>();**

**for (int num : arr) {**

**if (num >= 0) {**

**positive.add(num);**

**} else {**

**negative.add(num);**

**}**

**}**

**ArrayList<Integer> result = new ArrayList<>();**

**int i = 0, j = 0;**

**while (i < positive.size() && j < negative.size()) {**

**result.add(positive.get(i++));**

**result.add(negative.get(j++));**

**}**

**while (i < positive.size()) {**

**result.add(positive.get(i++));**

**}**

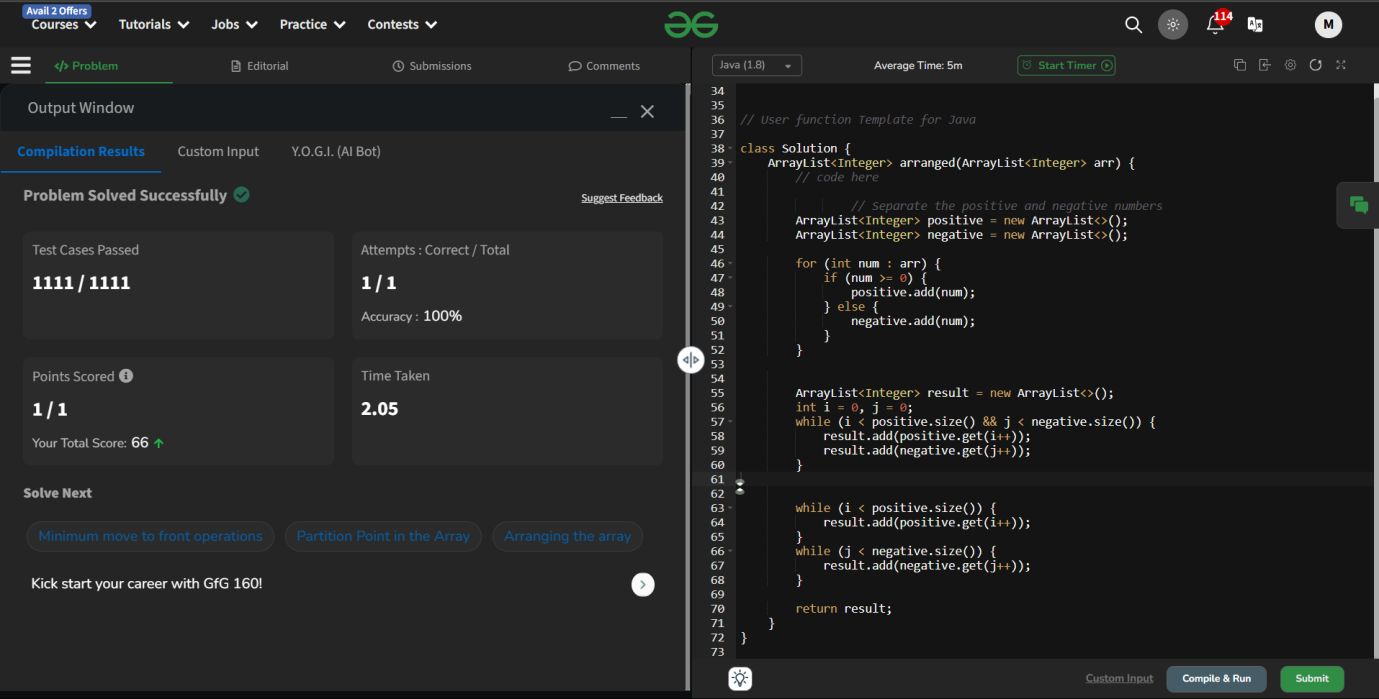
**while (j < negative.size()) {**

**result.add(negative.get(j++));**

**}**

**return result;**

**}**

**}**

**Q12. Searching a number**

**class Solution {**

**public int search(int k, ArrayList<Integer> arr) {**

**// code here**

**for ( int i = 0; i < arr.size(); i++){**

**if ( arr.get(i) == k){**

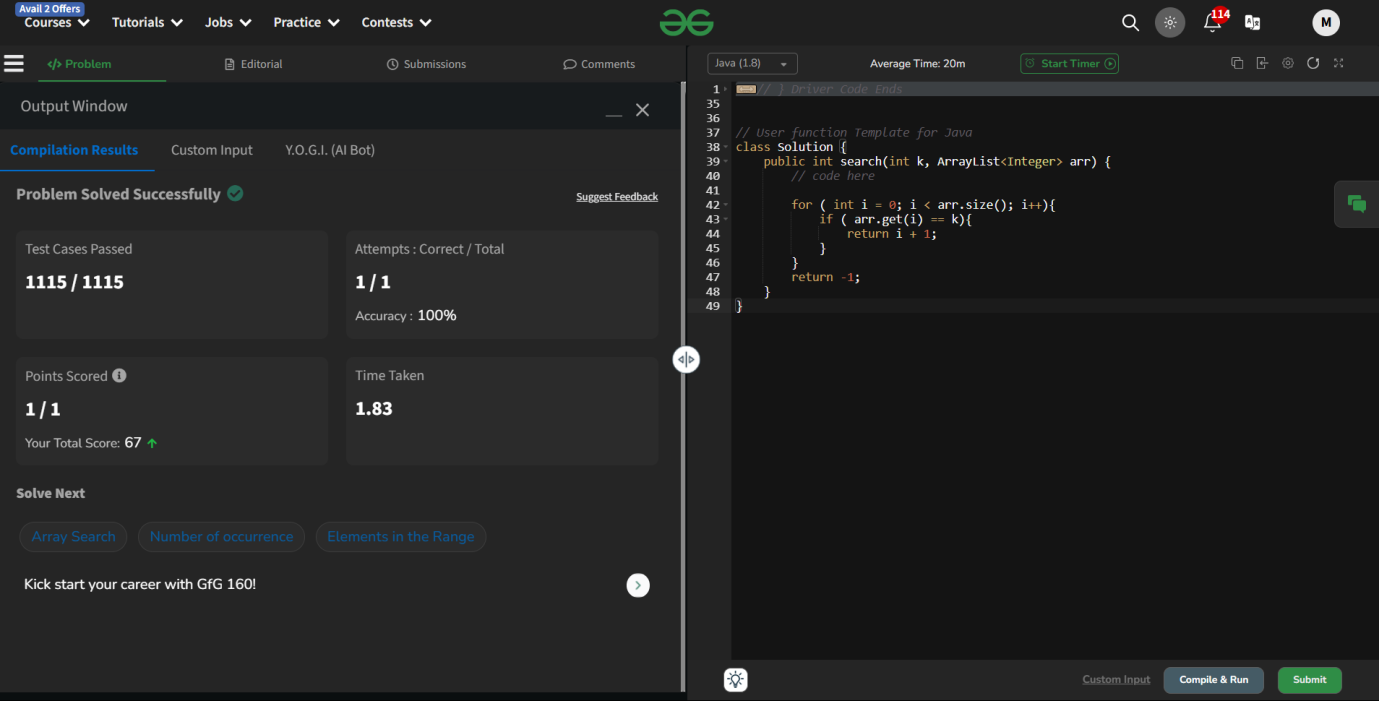
**return i + 1;**

**}**

**}**

**return -1;**

**}**

**}**

**Q13. Find unique element**

**class Solution {**

**public int findUnique(int k, int[] arr) {**

**// code here**

**int [] bitCount = new int[32];**

**for ( int num : arr){**

**for( int i = 0; i < 32 ; i++){**

**bitCount[i] += (num >> i) & 1;**

**}**

**}**

**int result = 0;**

**for ( int i = 0; i < 32 ; i++){**

**if ( bitCount[i] % k != 0){**

**result |= ( 1 << i);**

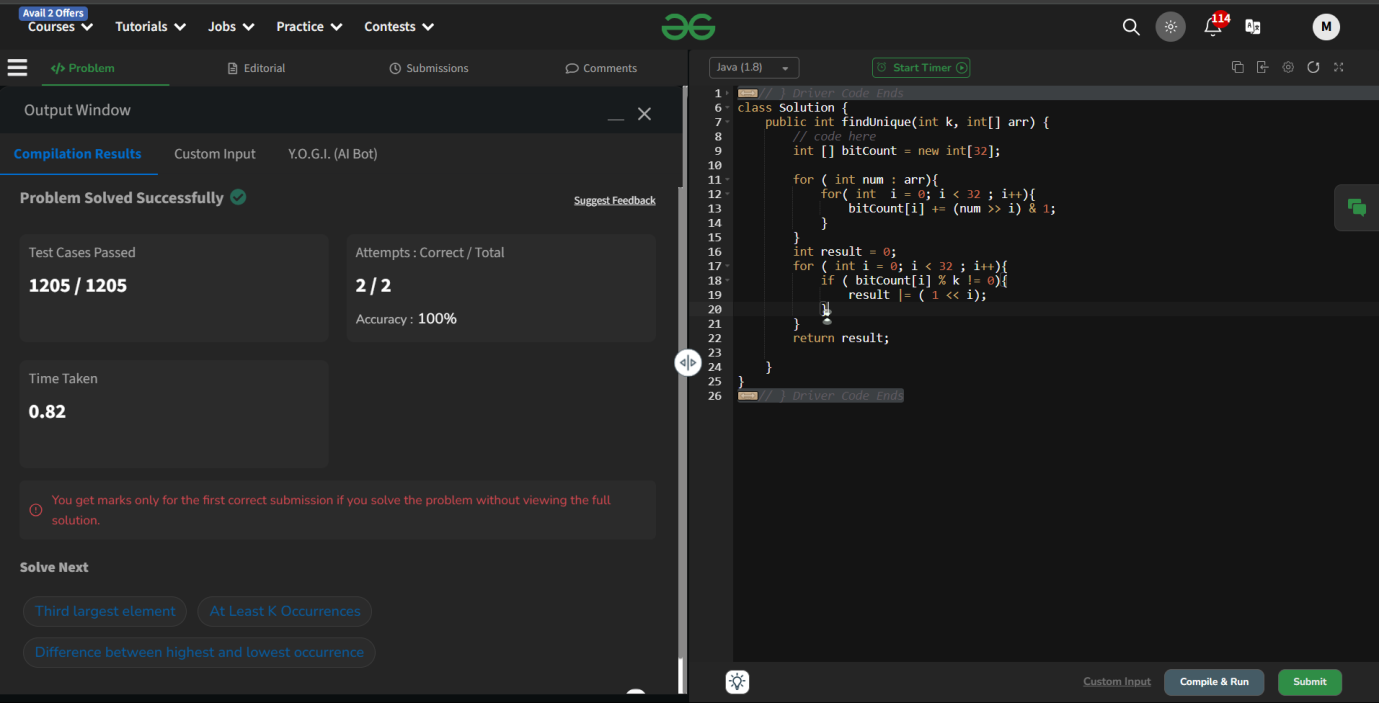
**}**

**}**

**return result;**

**}**

**}**

****

**Q14. Replace all 0's with 5**

**class Solution {**

**int convertfive(int num) {**

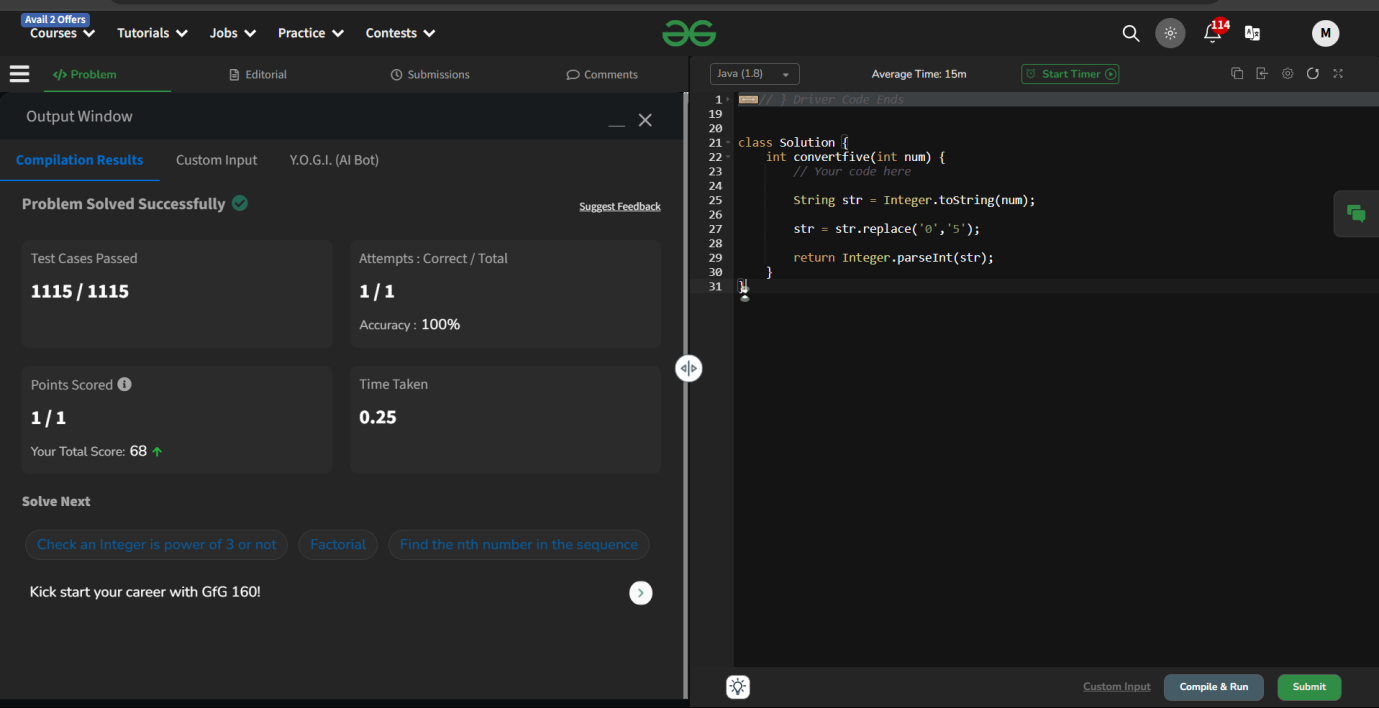
**// Your code here**

**String str = Integer.toString(num);**

**str = str.replace('0','5');**

**return Integer.parseInt(str);**

**}**

**}**

**Q15. Largest product**

**class Solution {**

**public int findMaxProduct(int[] arr, int k) {**

**// code here**

**if (k > arr.length) return -1;**

**int maxProduct = Integer.MIN\_VALUE;**

**for (int i = 0; i <= arr.length - k; i++) {**

**int product = 1;**

**for (int j = i; j < i + k; j++) {**

**product \*= arr[j];**

**}**

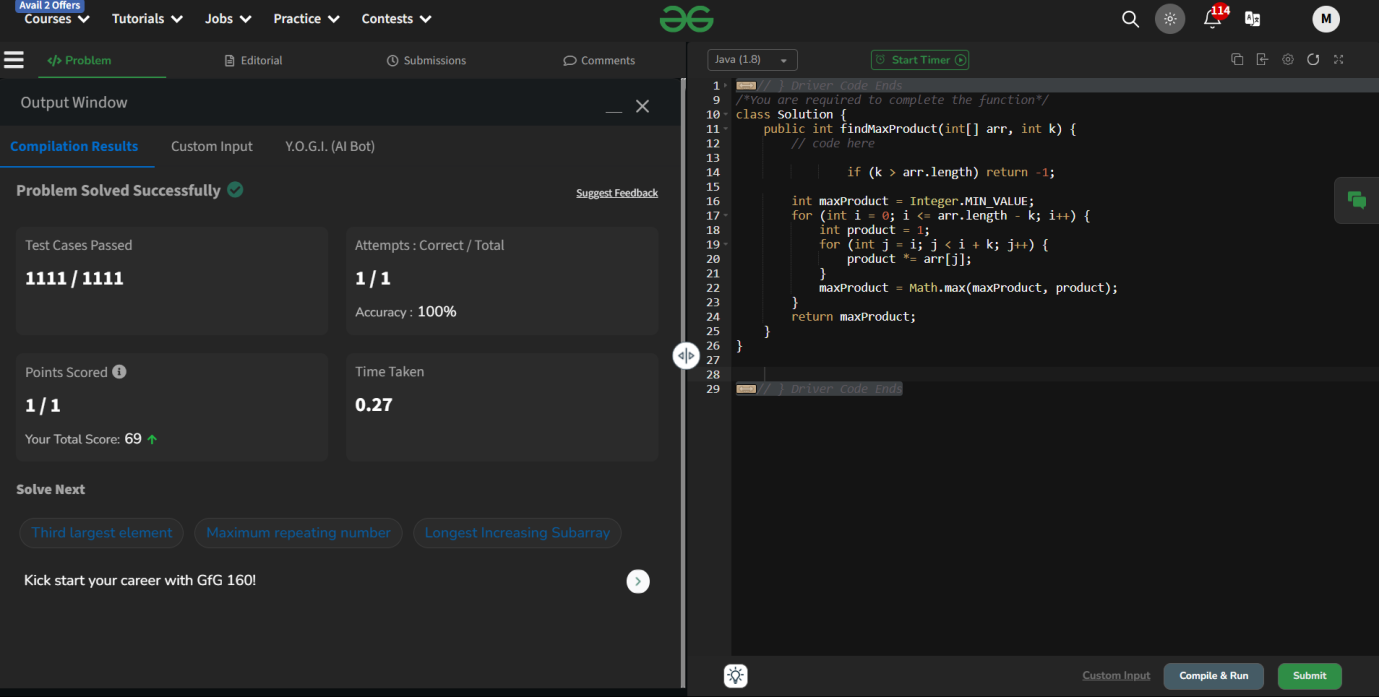
**maxProduct = Math.max(maxProduct, product);**

**}**

**return maxProduct;**

**}**

**}**

****

**Q16. Find element at a given Index**

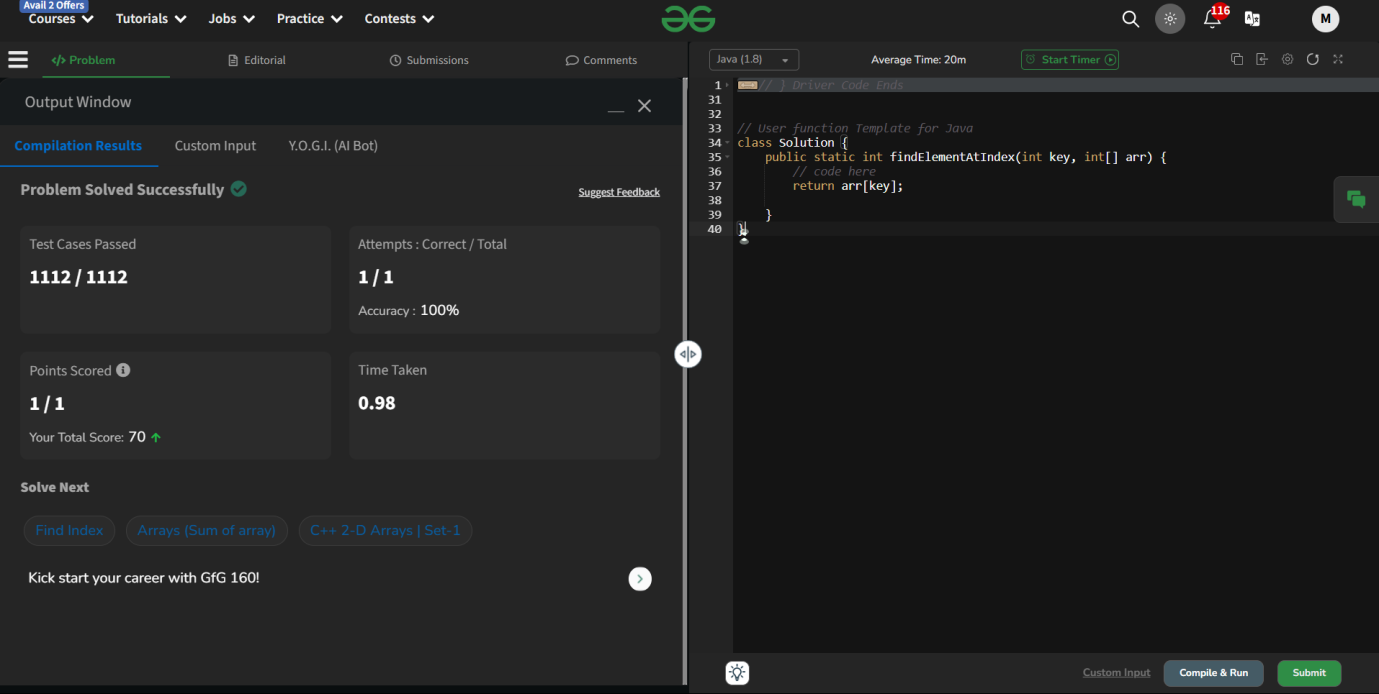
**class Solution {**

**public static int findElementAtIndex(int key, int[] arr) {**

**// code here**

**return arr[key];**

**}**

**}**

**Q17.Find unique element**

**class Solution {**

**public int findUnique(int k, int[] arr) {**

**// code here**

**int [] bitCount = new int[32];**

**for ( int num : arr){**

**for( int i = 0; i < 32 ; i++){**

**bitCount[i] += (num >> i) & 1;**

**}**

**}**

**int result = 0;**

**for ( int i = 0; i < 32 ; i++){**

**if ( bitCount[i] % k != 0){**

**result |= ( 1 << i);**

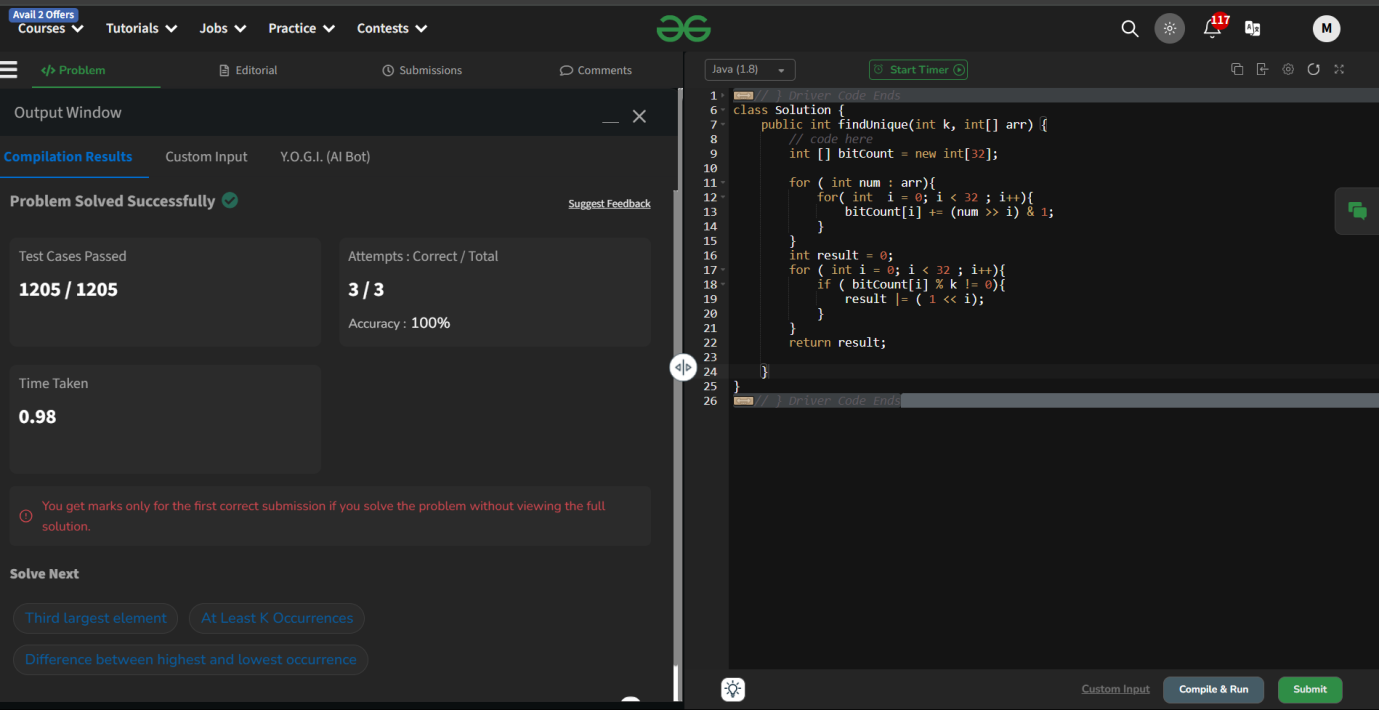
**}**

**}**

**return result;**

**}**

**}**

****

**Q8. Largest Element in Array**

**class Solution {**

**public static int largest(int[] arr) {**

**int max = 0;**

**for ( int i = 0; i<arr.length; i++){**

**if ( arr[i] > max){**

**max = arr[i];**

**}**

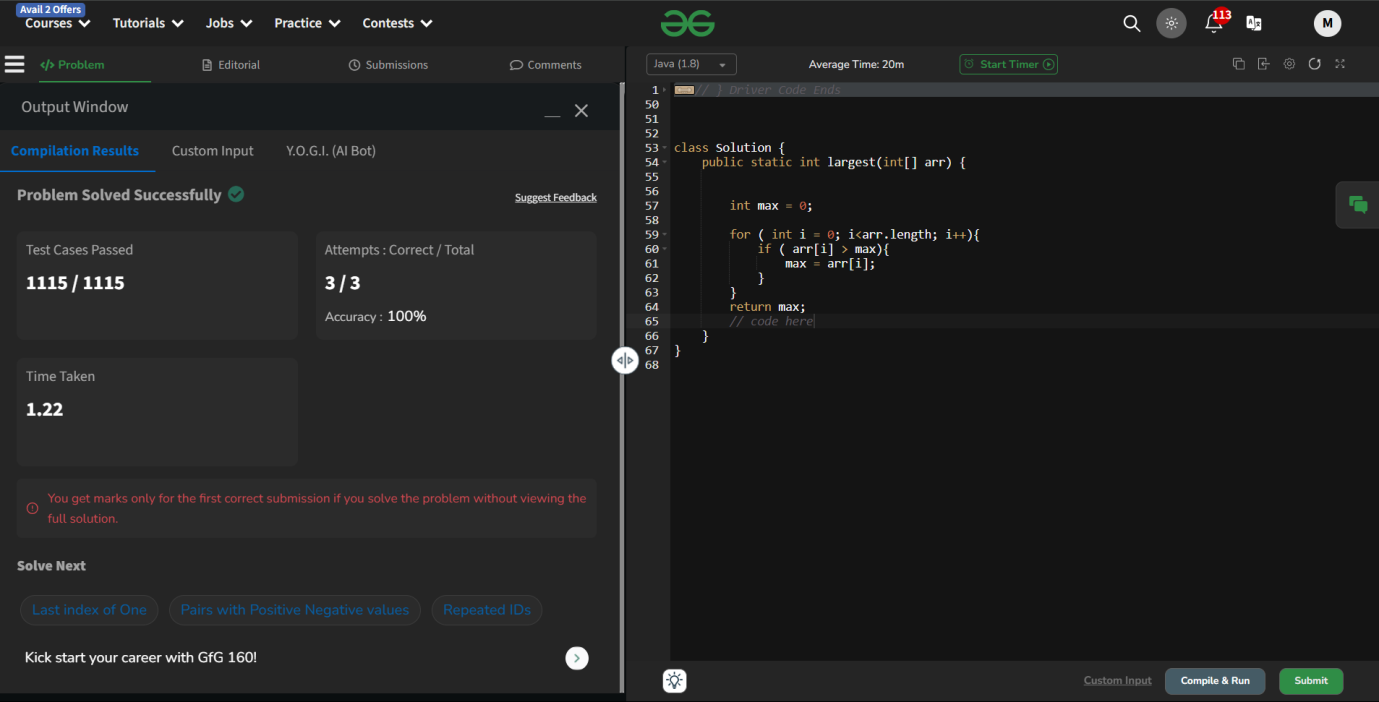
**}**

**return max;**

**// code here**

**}**

**}**

****

**Q19.Third largest element**

**class Solution {**

**int thirdLargest(int arr[]) {**

**// Your code here**

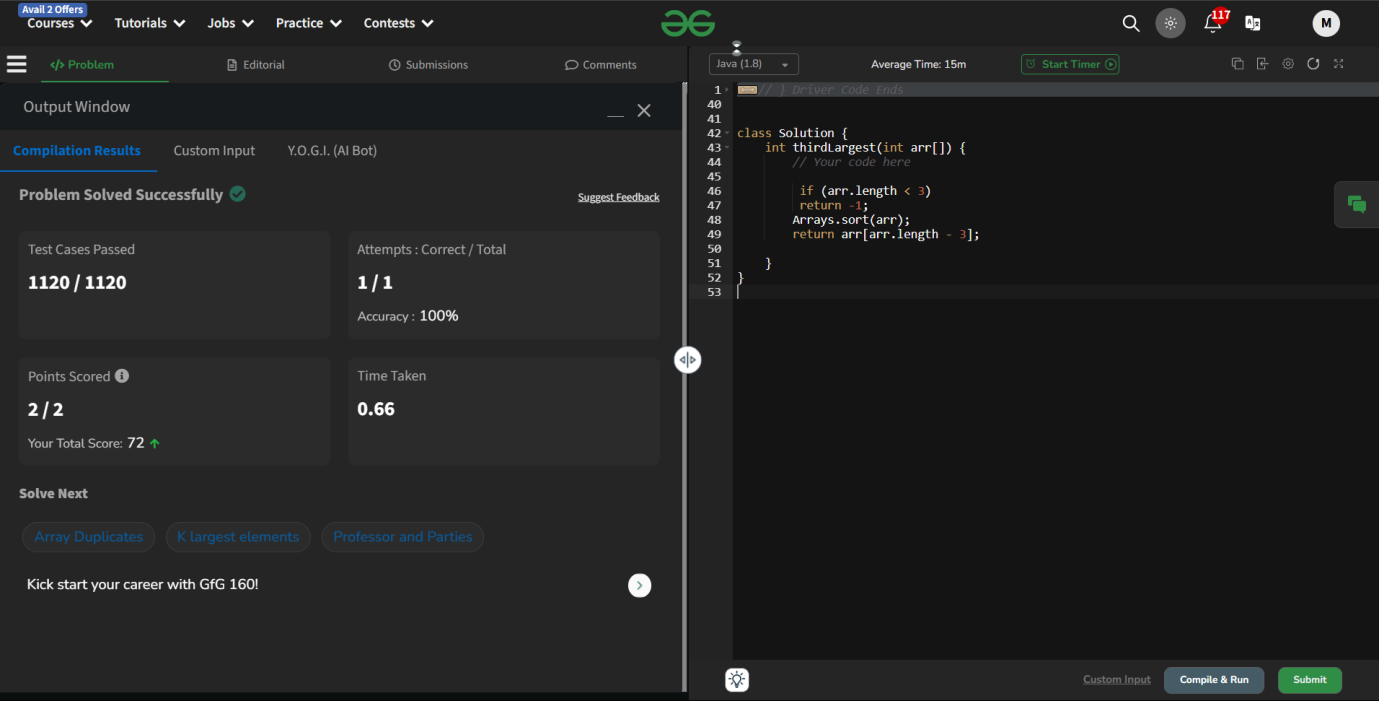
**if (arr.length < 3)**

**return -1;**

**Arrays.sort(arr);**

**return arr[arr.length - 3];**

**}**

**}**

**Q20. Display longest name**

**class Solution {**

**public String longest(List<String> arr) {**

**// code here**

**String longStr = "";**

**for ( String name : arr){**

**if ( name.length() > longStr.length()){**

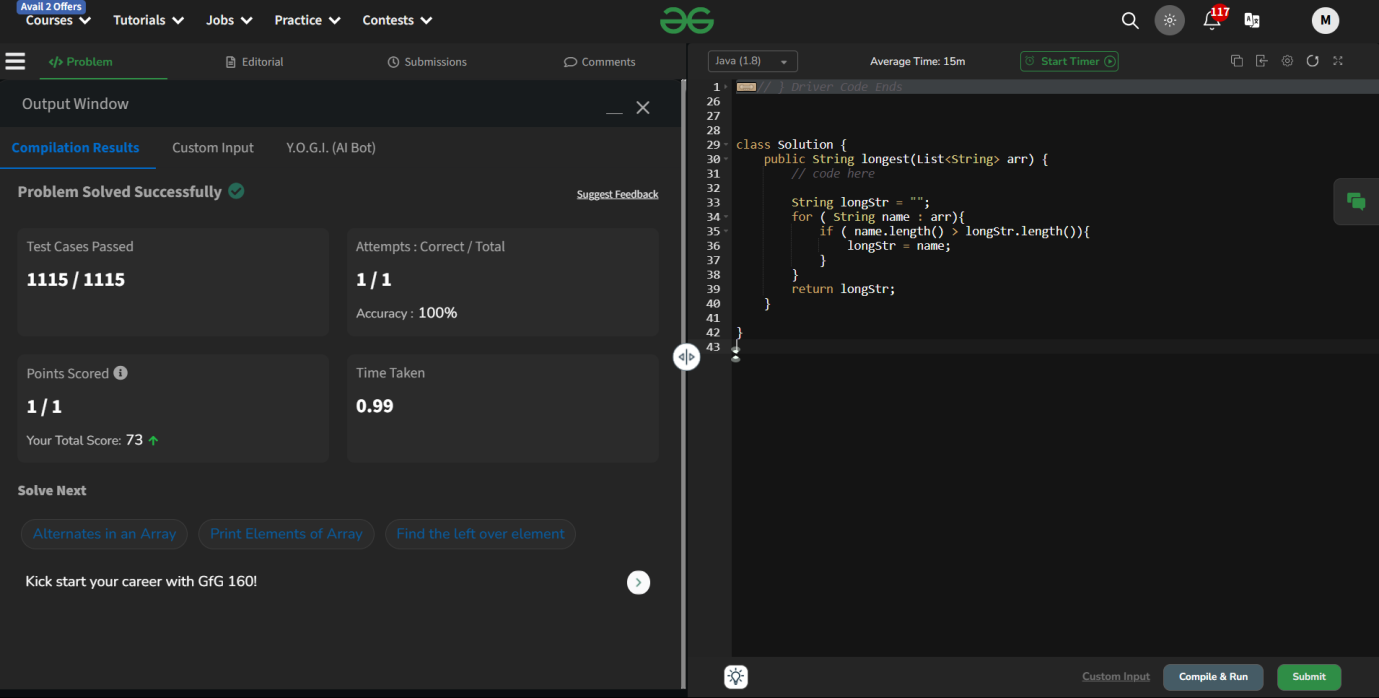
**longStr = name;**

**}**

**}**

**return longStr;**

**}**

**}**

**Q21.Array Search**

**class Solution {**

**static int search(int arr[], int x) {**

**for (int i = 0; i < arr.length; i++) {**

**if (arr[i] == x) {**

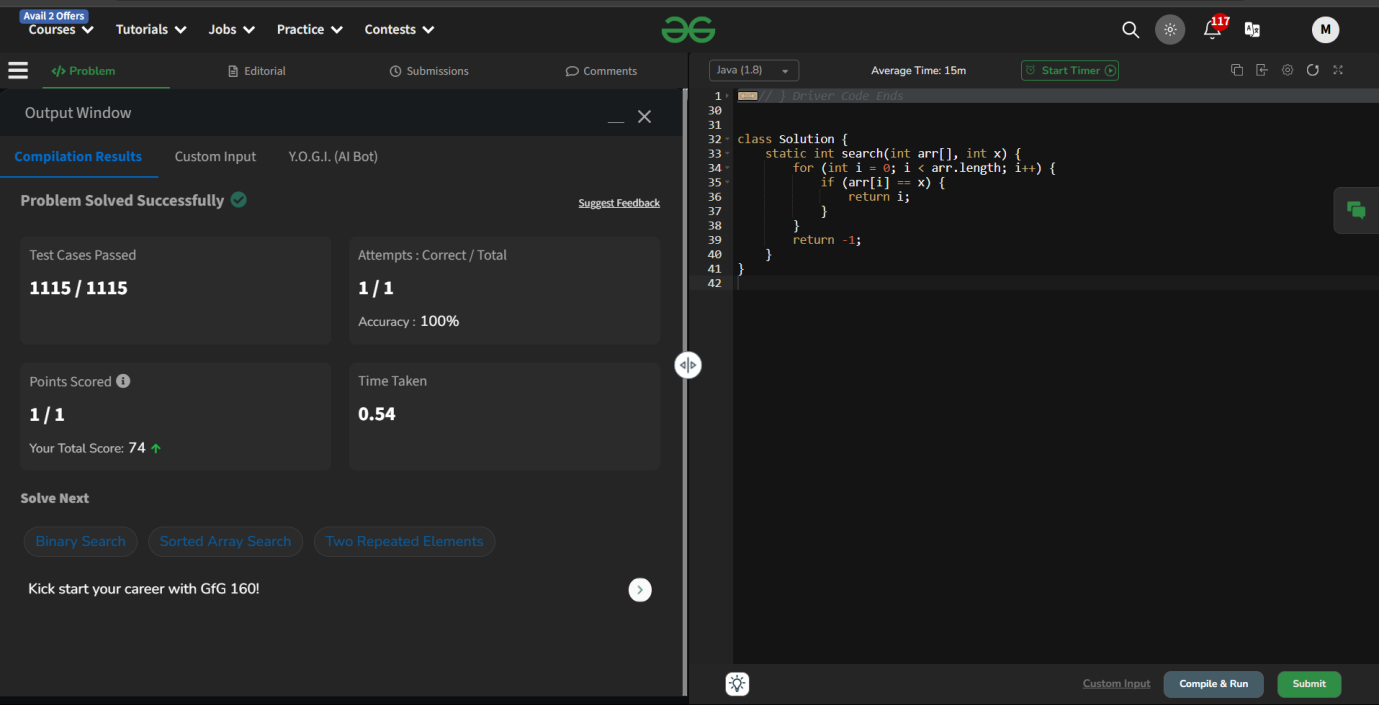
**return i;**

**}**

**}**

**return -1;**

**}**

**}**

**Q22.Merge and Sort**

**class Solution { public int[] mergeNsort(int[] arr, int[] brr) { Set<Integer> set = new TreeSet<>();**

**for (int num : arr)**

**set.add(num);**

**for (int num : brr)**

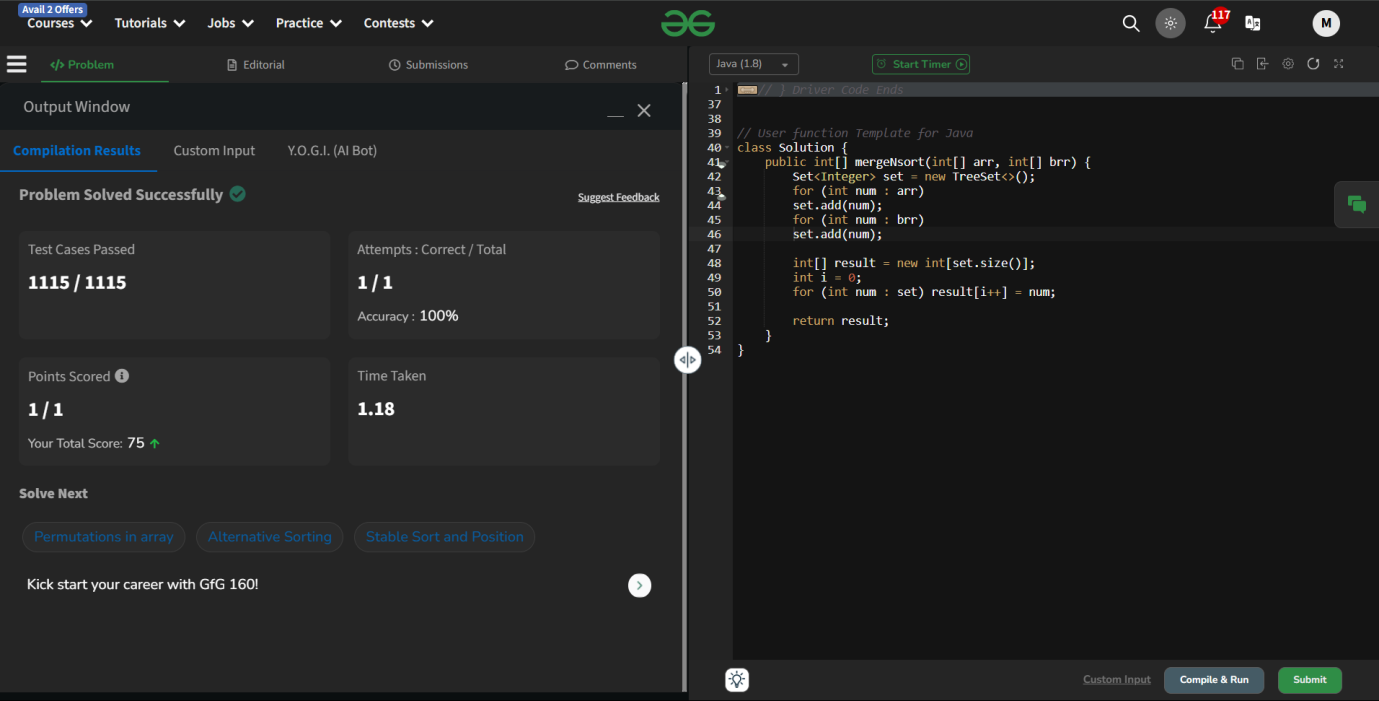
**set.add(num);**

**int[] result = new int[set.size()]; int i = 0;**

**for (int num : set) result[i++] = num; return result;**

**}**

**}**

****

**Q23.Maximum product of two numbers**

**class Solution {**

**public static int maxProduct(int[] arr) {**

**// code here**

**if (arr.length < 2) {**

**return -1; // Not enough elements to form a product**

**}**

**int max1 = Integer.MIN\_VALUE, max2 = Integer.MIN\_VALUE;**

**for (int num : arr) {**

**if (num > max1) {**

**max2 = max1;**

**max1 = num;**

**} else if (num > max2) {**

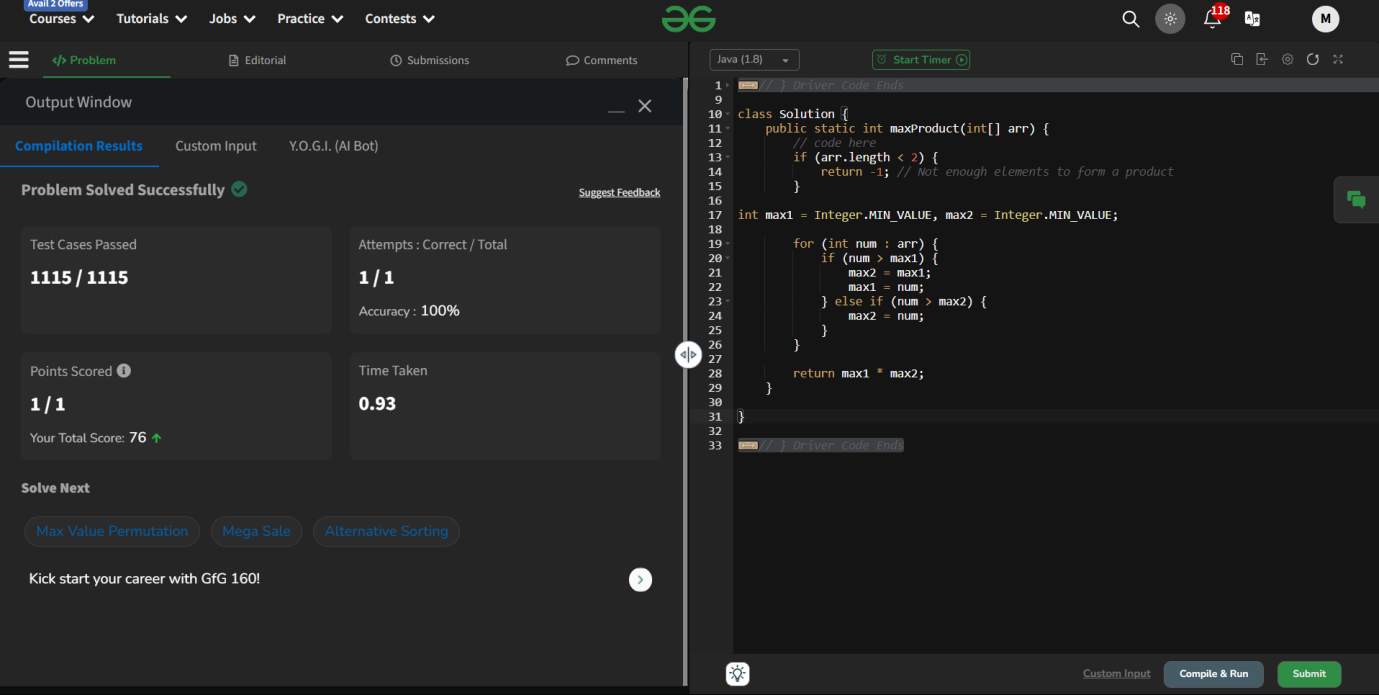
**max2 = num;**

**}**

**}**

**return max1 \* max2;**

**}**

**}**

**Q24. Difference between highest and lowest occurrence**

**class Solution {**

**// Function to find the difference between the maximum and minimum frequency of**

**// elements.**

**public int findDiff(int[] arr) {**

**// code here**

**HashMap<Integer, Integer> freqMap = new HashMap<>();**

**for (int num : arr) {**

**freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);**

**}**

**int maxFreq = 0, minFreq = Integer.MAX\_VALUE;**

**for (int count : freqMap.values()) {**

**if (count > maxFreq) {**

**maxFreq = count;**

**}**

**if (count < minFreq) {**

**minFreq = count;**

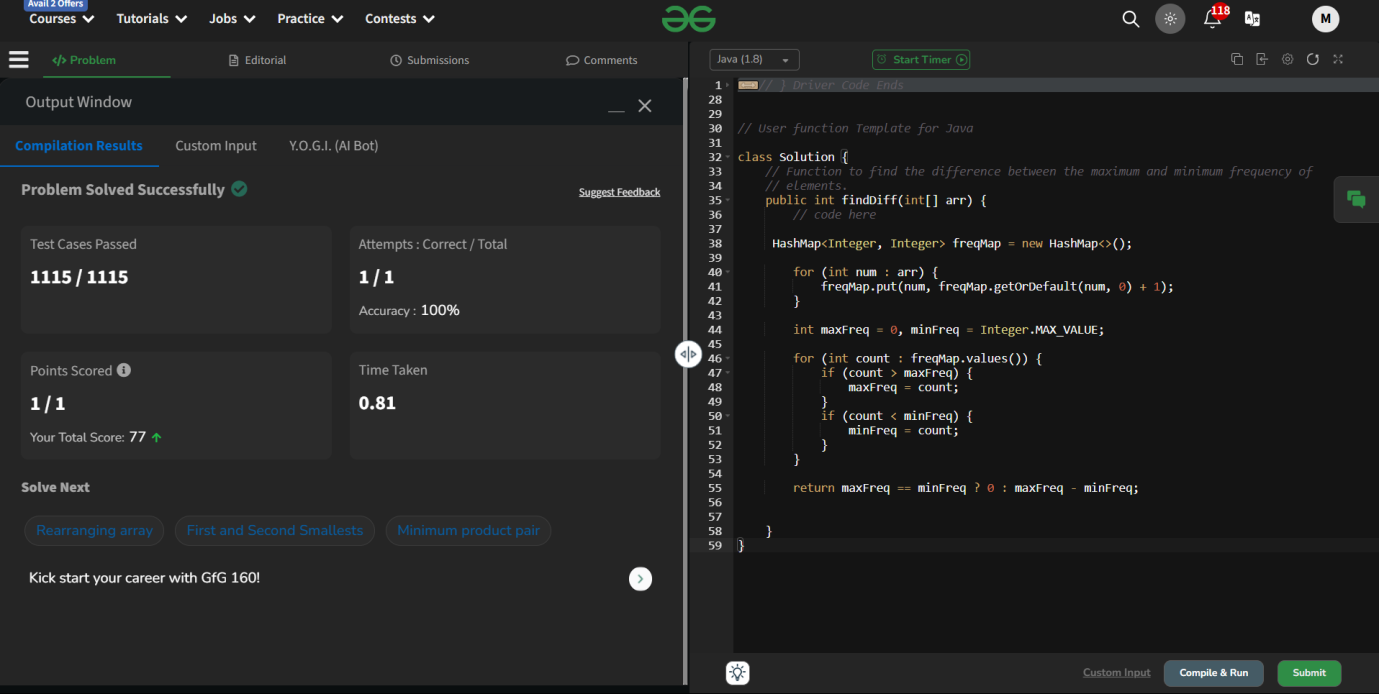
**}**

**}**

**return maxFreq == minFreq ? 0 : maxFreq - minFreq;**

**}**

**}**

****

**Q25.Last index of One**

**class Solution {**

**public int lastIndex(String s) {**

**return s.lastIndexOf('1');**

**}**

**public static void main(String[] args) {**

**Solution sol = new Solution();**

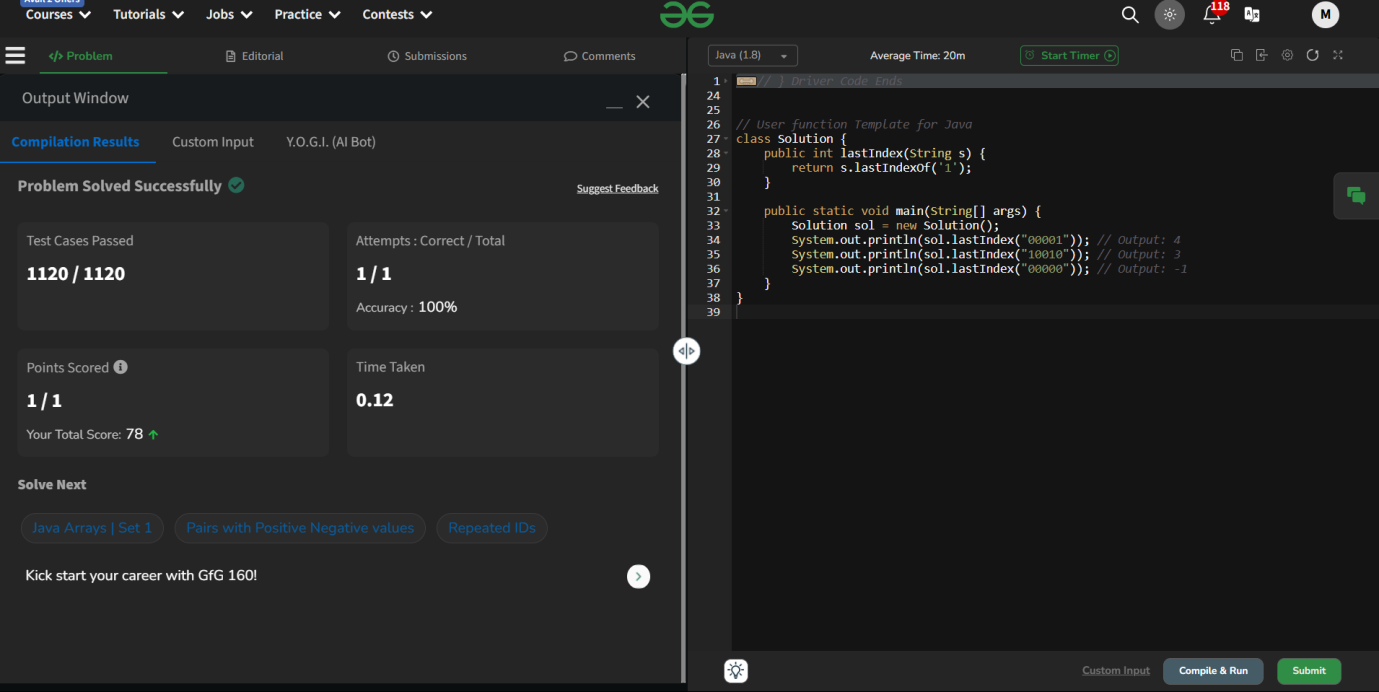
**System.out.println(sol.lastIndex("00001")); // Output: 4**

**System.out.println(sol.lastIndex("10010")); // Output: 3**

**System.out.println(sol.lastIndex("00000")); // Output: -1**

**}**

**}**

****

**Q26. Palindromic Array**

**class Solution {**

**public static boolean isPalinArray(int[] arr) {**

**// add code here.**

**// bu using for each loop**

**for ( int num : arr){**

**if ( !isPalindrome(num)){**

**return false;**

**}**

**}**

**return true;**

**}**

**private static boolean isPalindrome(int num){**

**int original = num ;**

**int reverse = 0;**

**while ( num > 0){**

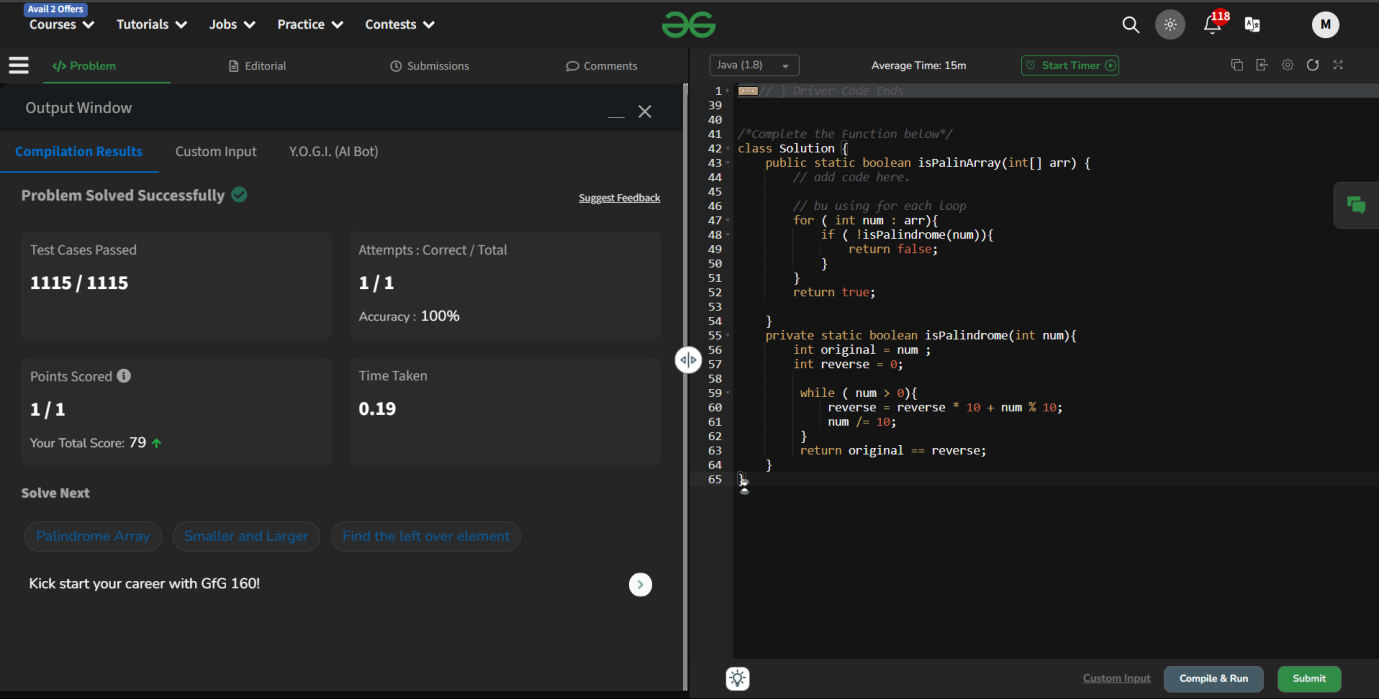
**reverse = reverse \* 10 + num % 10;**

**num /= 10;**

**}**

**return original == reverse;**

**}**

**}**

**Q29. Sum of Array**

**class Solution {**

**int arraySum(int arr[]) {**

**// code here**

**int sum = 0;**

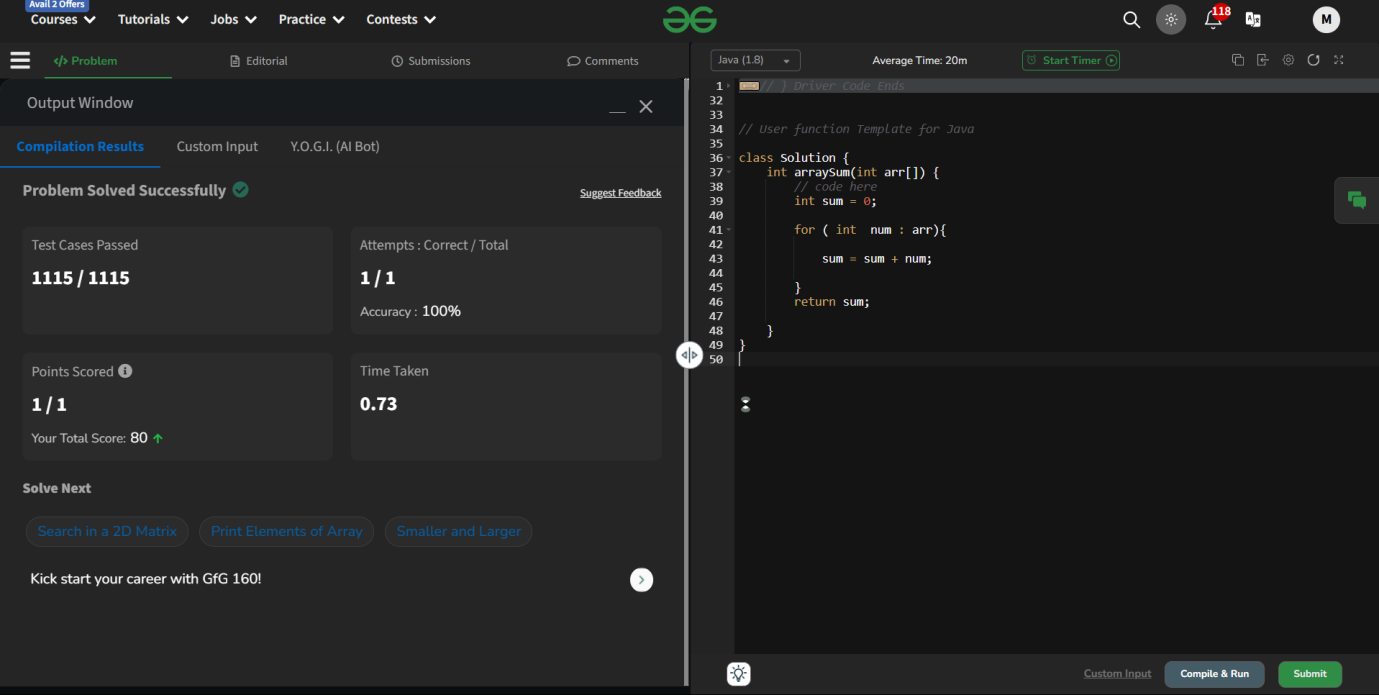
**for ( int num : arr){**

**sum = sum + num;**

**}**

**return sum;**

**}**

**}**