

SRI VASAVI ENGINEERING COLLEGE (Autonomous)
PEDATADEPALLI, TADEPALLIGUDEM.



Certificate

*This is to certify that this is a bonafide record of Practical Work done in **Master Coding in Competitive Programming-I Lab** by Mr / Ms _____ bearing Roll No. _____ of _____ Branch of V Semester during the academic year 2025-26.*

No. of Experiments Done:

Faculty Incharge of the Laboratory

Head of the Department

EXTERNAL EXAMINER

Screenshot of a web browser showing a user profile page on GeeksforGeeks.org. The URL in the address bar is geeksforgeeks.org/user/moturiprasad08/. The page displays the user's profile information, including their name (moturiprasad08), institution (Sri Vasavi Engineering College (SVEC) Tadepalligudem), language used (Java, C++), rank (522), coding score (772), problems solved (419), and contest rating (—). The sidebar on the left shows links for Profile, Contributions, Saved Items, Colleges, Companies, Campus Mantri, Invite, Edit Profile, and Account Settings. A banner at the top offers a 40% discount on GfG Courses + Mock Interviews.

Profile

Contributions

Saved Items

Colleges

Companies

Campus Mantri

Invite

Edit Profile

Account Settings

Search...

Switch to Dark Mode

Profile

moturiprasad08

Institution
Sri Vasavi Engineering College
(SVEC) Tadepalligudem

Language Used
Java, C++

Rank 522

Current POTD Streak 01/1537 days

Coding Score 772

Problem Solved 419

Contest Rating —

Campus Mantri
pavana26

455 submissions in current year

Current

The screenshot shows a LeetCode profile page for user 'im_prasad'. The profile summary indicates a rank of 1,854,178 and 82 solved problems out of 3716 total submissions. The user has 0 badges. A circular progress bar shows 82 solved problems. Below the summary, there's a 'Community Stats' section with 0 views, 0 solutions, 0 discussions, and 0 reputation. The 'Languages' section shows Java with 82 problems solved. The 'Skills' section lists 'Advanced' and 'Dynamic Programming' with 8x solved problems. A recent activity feed shows three submissions: 'Swap Nodes in Pairs' (2 minutes ago), 'Generate Parentheses' (4 minutes ago), and 'Merge Two Sorted Lists' (4 minutes ago). The page is displayed in a browser window with other tabs visible at the top.

EXP.NO	NAME OF THE EXPERIMENT	Page No.	SIGNATURE
Programs on Mathematical problems			
1	Sum of series	07	
2	Armstrong number	08	
3	Reverse the digits	09	
4	Count of Total numbers	10-11	
5	Power of Another	12	
6	Angle Between hour and minute hand	13	
7	In Sequence	14	
8	Sum of GCDs	15	
9	Juggler Sequence	16	
10	Kth Digit	17	
Programs on Array problems			
11	Two Sum	18	
12	Single Number	19	
13	Contains Duplicate	20	
14	Move Zeroes	21	
15	Missing Number	22	
16	Second Largest	23	
17	Array Leader	24	
18	Find Duplicates	25	
19	Sort 0 1 2	26	
20	Binary Search	27	
Programs on Fibonacci Problems			
21	Nth Fibonacci Number	28	
22	Fibonacci Series	29-30	
23	Check if the Number is Fibonacci	31	
24	Fibonacci Number	32	

Programs on String Problems

25	Palindrome	33	
26	Reverse the String	34	
27	Reverse Words	35	
28	Non Repeating Character	36	
29	Rotated String	37	
30	Remove Duplicates	38-39	
31	Max Occuring Character	40	
32	Last Index	41	
33	Anagram	42	
34	Remove Consecutive Duplicate Character	43	

Programs on Series Problems

35	Juggler Sequence	44-45	
36	Nth of Series	46	
37	Difference Series	47	
38	Deficiency	48-49	

Programs on Prime Numbers Problems

39	Prime Number	50	
40	Largest Prime Factors	51-52	
41	Unique Prime Factor	53-54	
42	Get prime	55	
43	Prime Range	56-57	
44	Is Prime	58	

Programs on Pattern Problems

45	Sequence of numbers	59	
46	Isosceles triangle of stars	60	

Programs on Matrices Problems			
47	Row-wise sorted matrices	61-62	
48	Transpose of the Matrix	63	
49	2D Dimensions	64	
50	Square Matrix	65	

Mathematical Problems

1. Given an integer **n**, calculate the sum of series $1^3 + 2^3 + 3^3 + 4^3 + \dots$ till n-th term.
// User function Template for Java

```
class Solution {  
    int sumOfSeries(int n) {  
        // Using the formula: sum = (n * (n + 1) / 2) ^ 2  
        int sum = (n * (n + 1)) / 2;  
        return sum * sum;  
    }  
}
```

OUTPUT:

The screenshot shows a browser window with multiple tabs open. The active tab is 'geeksforgeeks.org/problems/sum-of-first-n-terms5843/1?page=1&category=Mathematical&sortBy=submissions'. The page displays a Java code editor and an output window.

Java Code:

```
1 // User function Template for Java  
2  
3 class Solution {  
4     int sumOfSeries(int n) {  
5         // Using the formula: sum = (n * (n + 1) / 2) ^ 2  
6         int sum = (n * (n + 1)) / 2;  
7         return sum * sum;  
8     }  
9 }  
10 }
```

Output Window:

Compilation Results Custom Input

Compilation Completed

Case 1

Input: 5

n = 5

Your Output:
225

Expected Output:
225

Custom Input Compile & Run Submit

2. You are given a **3-digit** number **n**, Find whether it is an **Armstrong** number or not.

// User function Template for Java

```
class Solution {
```

```
    static boolean armstrongNumber(int n) {
```

```
        int original = n;
```

```
        int sum = 0;
```

```
        while (n > 0) {
```

```
            int digit = n % 10;
```

```
            sum += digit * digit * digit; // cube of digit
```

```
            n /= 10;
```

```
}
```

```
        return sum == original;
```

```
}
```

}**OUTPUT:**

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The URL in the address bar is geeksforgeeks.org/problems/armstrong-numbers2727/1?page=1&category=Mathematical&sortBy=submissions. The page title is "geeksforgeeks.org/problems/armstrong-numbers2727/1?page=1&category=Mathematical&sortBy=submissions".

The main content area displays the Java code for finding Armstrong numbers. The code is as follows:

```
1 // User function Template for Java
2 class Solution {
3     static boolean armstrongNumber(int n) {
4         int original = n;
5         int sum = 0;
6
7         while (n > 0) {
8             int digit = n % 10;
9             sum += digit * digit * digit; // cube of digit
10            n /= 10;
11        }
12
13        return sum == original;
14    }
15
16 }
```

The "Output Window" tab is active, showing the results of the code execution. It includes sections for "Compilation Results" and "Custom Input".

Compilation Completed

Case 1

Input: n = 153

Your Output: true

Expected Output: true

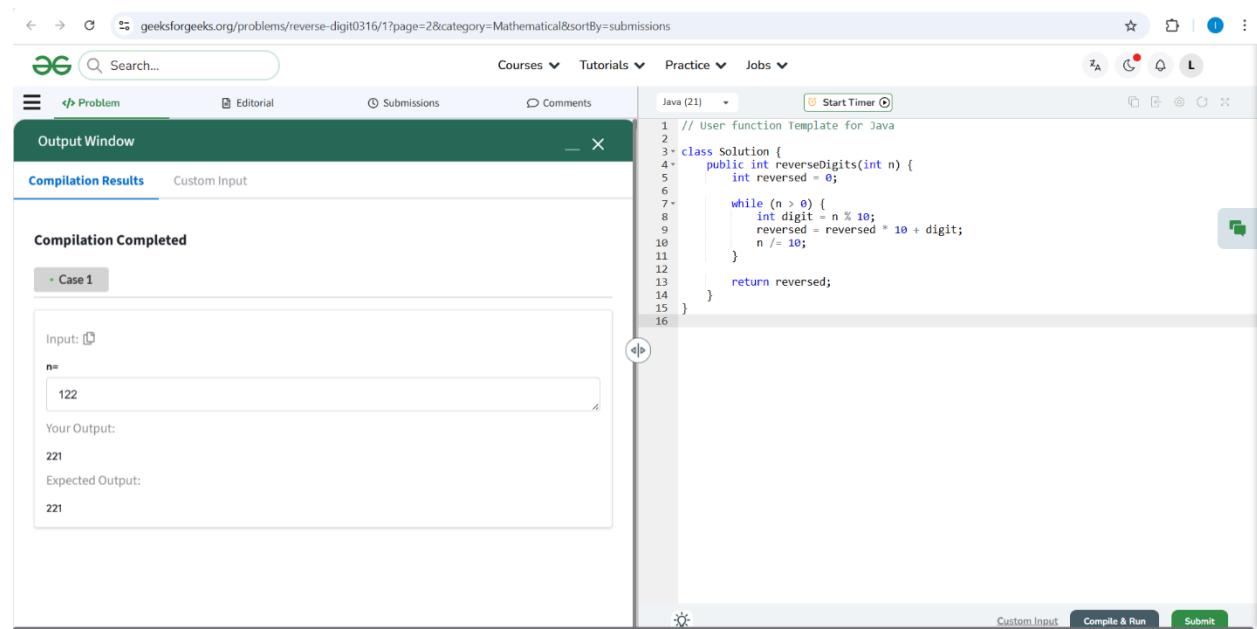
At the bottom right of the window, there are buttons for "Custom Input", "Compile & Run", and "Submit".

3. You are given an integer **n**. Your task is to reverse the digits, ensuring that the reversed number has no leading zeroes.

```
// User function Template for Java
```

```
class Solution {  
    public int reverseDigits(int n) {  
        int reversed = 0;  
  
        while (n > 0) {  
            int digit = n % 10;  
            reversed = reversed * 10 + digit;  
            n /= 10;  
        }  
  
        return reversed;  
    }  
}
```

OUTPUT:



The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is [geeksforgeeks.org/problems/reverse-digit0316/1?page=2&category=Mathematical&sortBy=submissions](https://www.geeksforgeeks.org/problems/reverse-digit0316/1?page=2&category=Mathematical&sortBy=submissions). The page title is "Output Window". The main content area displays the Java code for reversing digits. The code is pasted into a code editor window. The code itself is:

```
1 // User function Template for Java  
2  
3 class Solution {  
4     public int reverseDigits(int n) {  
5         int reversed = 0;  
6  
7         while (n > 0) {  
8             int digit = n % 10;  
9             reversed = reversed * 10 + digit;  
10            n /= 10;  
11        }  
12  
13        return reversed;  
14    }  
15 }
```

Below the code editor, there is an "Output Window" section. It shows the input "122" and the output "221". The status bar at the bottom indicates "Custom Input" and "Compile & Run" buttons.

4. You are given a number **n**, Return the count of total numbers from **1** to **n** containing **4** as a digit.

```
class Solution {  
  
    int countNumberswith4(int n) {  
  
        int count = 0;  
  
        for (int i = 1; i <= n; i++) {  
  
            if (containsFour(i)) {  
  
                count++;  
  
            }  
  
            return count;  
        }  
  
        boolean containsFour(int num) {  
  
            while (num > 0) {  
  
                if (num % 10 == 4) {  
  
                    return true;  
                }  
  
                num /= 10;  
            }  
  
            return false;  
        }  
    }  
}
```

OUTPUT:

The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is a solution for a problem that counts how many numbers between 1 and n contain the digit 4. It uses a helper function `containsFour` to check if a number contains the digit 4.

```
Java (21) Start Timer
1 class Solution {
2     int countNumbersWith4(int n) {
3         int count = 0;
4         for (int i = 1; i <= n; i++) {
5             if (containsfour(i)) {
6                 count++;
7             }
8         }
9         return count;
10    }
11 }
12 boolean containsFour(int num) {
13     while (num > 0) {
14         if (num % 10 == 4) {
15             return true;
16         }
17         num /= 10;
18     }
19     return false;
20 }
21 }
22 }
```

The code editor interface includes tabs for 'Output Window', 'Compilation Results', and 'Custom Input'. The 'Compilation Results' tab shows 'Compilation Completed' with one case labeled 'Case 1'. The input is '9' and the output is '1', which matches the expected output.

5. Given two positive integers x and y , determine if y is a power of x . If y is a power of x , return **True**. Otherwise, return **False**.

```
class Solution {  
    public boolean isPowerOfAnother(int x, int y) {  
        if (x == 1) {  
            return y == 1;  
        }  
        while (y % x == 0) {  
            y = y / x;  
        }  
        return y == 1;  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. On the left, the 'Output Window' shows the 'Compilation Results' section, which indicates 'Compilation Completed'. Below it, the 'Case 1' input and output fields are shown. The input field contains '2 8', and the output field shows 'True'. At the bottom of the interface, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1+ class Solution {  
2+     public boolean isPowerOfAnother(int x, int y) {  
3+         if (x == 1) {  
4+             return y == 1;  
5+         }  
6+         while (y % x == 0) {  
7+             y = y / x;  
8+         }  
9+         return y == 1;  
10+    }  
11+ }  
12+ }
```

6. Given a string s representing time in 24-hour format "HH:MM", compute the **smallest angle in degrees** between the hour and minute hands of an analog clock.

```
class Solution {  
    public double getAngle(String s) {  
        String[] parts = s.split(":");  
        int hour = Integer.parseInt(parts[0]);  
        int minute = Integer.parseInt(parts[1]);  
        hour = hour % 12;  
        double minuteAngle = minute * 6;  
        double hourAngle = (hour * 30) + (minute * 0.5);  
        double diff = Math.abs(hourAngle - minuteAngle);  
        return Math.min(diff, 360 - diff);  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is geeksforgeeks.org/problems/angle-between-hour-and-minute-hand0545/1?page=4&category=Mathematical&sortBy=submissions. The page title is "Angle Between Hour And Minute Hand".

The interface includes a navigation bar with links for Courses, Tutorials, Practice, and Jobs. Below the navigation bar, there are tabs for Problem, Editorial, Submissions, and Comments. The "Output Window" tab is selected, showing the "Compilation Results" section which displays "Compilation Completed".

In the main workspace, the Java code for the solution is pasted into the "Java (21)" code editor. The code is identical to the one provided in the question. To the right of the code editor, there is a "Start Timer" button.

On the left side of the workspace, there is an "Input" field containing "s = 06:00". Below it, the "Your Output:" field shows "180.000" and the "Expected Output:" field also shows "180.000".

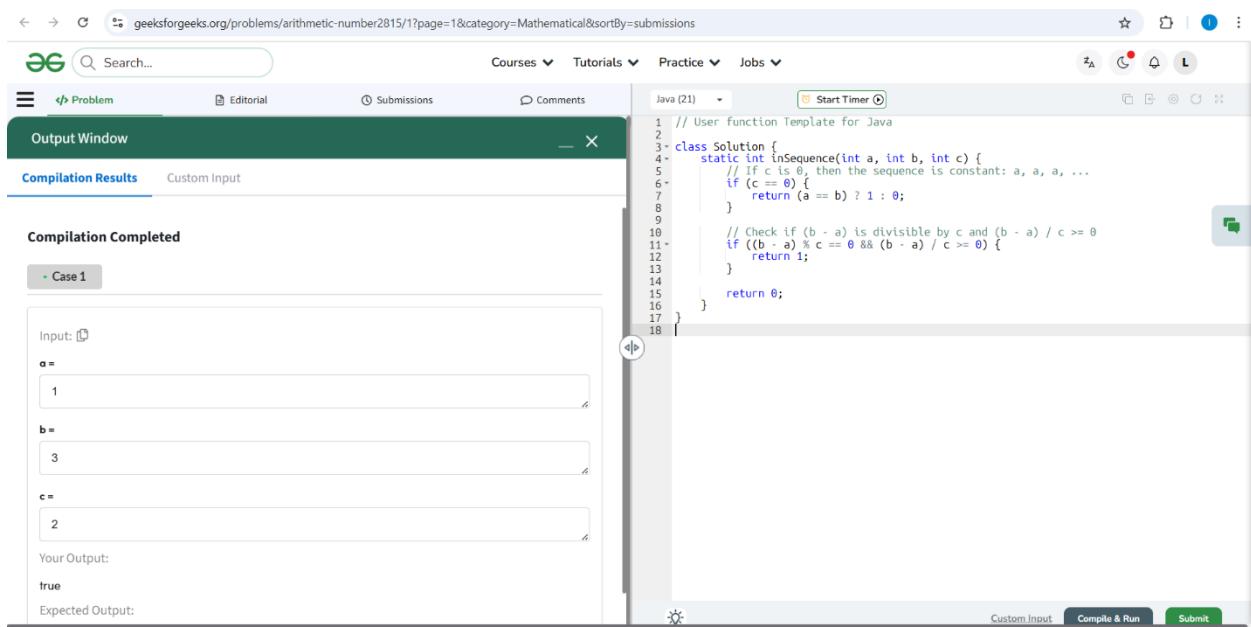
At the bottom of the workspace, there are buttons for "Custom Input", "Compile & Run", and "Submit".

7. Given three integers 'a' denotes the first term of an arithmetic sequence, 'c' denotes the common difference of an arithmetic sequence and an integer 'b'. you need to tell whether 'b' exists in the arithmetic sequence or not. Return 1 if b is present in the sequence. Otherwise, returns 0.

// User function Template for Java

```
class Solution {  
    static int inSequence(int a, int b, int c) {  
        // If c is 0, then the sequence is constant: a, a, a, ...  
        if (c == 0) {  
            return (a == b) ? 1 : 0;  
        }  
  
        // Check if (b - a) is divisible by c and (b - a) / c >= 0  
        if ((b - a) % c == 0 && (b - a) / c >= 0) {  
            return 1;  
        }  
  
        return 0;  
    }  
}
```

OUTPUT:



The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution, which is identical to the one provided above. Below the code is the 'Output Window' section, which contains the 'Compilation Results' tab. It shows 'Compilation Completed' and 'Case 1' results. The 'Input' fields show values: a = 1, b = 3, and c = 2. The 'Your Output:' field contains 'true', and the 'Expected Output:' field also contains 'true'. At the bottom of the window are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

8. Given two positive integers **a** and **b**, find GCD of **a** and **b**.

```
class Solution {  
    public static int gcd(int a, int b) {  
        // Euclidean algorithm: keep replacing a with b, and b with a % b  
        while (b != 0) {  
            int temp = b;  
            b = a % b;  
            a = temp;  
        }  
        return a;  
    }  
}
```

}OUTPUT:

The screenshot shows a Java development environment on the GeeksforGeeks website. The code in the editor is identical to the one above. In the 'Output Window' tab, under 'Compilation Results', it says 'Compilation Completed'. Under 'Case 1', the 'Input:' section shows 'a =' followed by '3' and 'b =' followed by '6'. The 'Your Output:' section shows '3', which matches the 'Expected Output:' shown below it. At the bottom of the window are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

9. Juggler Sequence is a series of integers in which the first term starts with a positive integer number a and the remaining terms are generated from the immediate previous term using the below recurrence relation:

```
import java.util.*;
class Solution {
    public ArrayList<Long> jugglerSequence(long n) {
        ArrayList<Long> seq = new ArrayList<>();
        seq.add(n);

        while (n != 1) {
            if (n % 2 == 0) {
                n = (long) Math.floor(Math.sqrt(n));
            } else {
                n = (long) Math.floor(Math.pow(n, 1.5));
            }
            seq.add(n);
        }
        return seq;
    }
}
```

OUTPUT:

The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is identical to the one provided above. In the 'Output Window' tab, under 'Compilation Results', it says 'Compilation Completed'. Below that, there's a 'Case 1' section with an input field containing '9'. The 'Your Output:' field shows the sequence '9 27 140 1136 621'. The 'Expected Output:' field also shows '9 27 140 1136 621'. At the bottom right of the editor, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

10. Given two numbers **a** and **b**, find **kth** digit from right of a^b .

```
import java.math.BigInteger;
class Solution {
    public int kthDigit(int a, int b, int k) {
        BigInteger base = BigInteger.valueOf(a);
        BigInteger result = base.pow(b);
        String str = result.toString();
        if (k > str.length()) {
            return -1;
        }
        char ch = str.charAt(str.length() - k);
        return ch - '0';
    }
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. Below the code, the 'Output Window' section shows the 'Compilation Completed' status. The 'Case 1' tab is selected, displaying the input values **a**: 3, **b**: 3, and **k**: 1. The 'Your Output:' field contains the value 7, which matches the 'Expected Output' shown below it. At the bottom of the window are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
import java.math.BigInteger;
class Solution {
    public int kthDigit(int a, int b, int k) {
        BigInteger base = BigInteger.valueOf(a);
        BigInteger result = base.pow(b);
        String str = result.toString();
        if (k > str.length()) {
            return -1;
        }
        char ch = str.charAt(str.length() - k);
        return ch - '0';
    }
}
```

Programs on Array problems

11. Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

```
import java.util.*;
class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];
            if (map.containsKey(complement)) {
                return new int[]{map.get(complement), i};
            }
            map.put(nums[i], i);
        }
        return new int[]{};
    }
}
```

OUTPUT:

The screenshot shows a LeetCode problem submission page for the "Two Sum" problem. The code is identical to the one above. The submission was accepted with a runtime of 0 ms. The input and output fields show the expected values for the provided test cases.

Code:

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

Testcase: Accepted Runtime: 0 ms

Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:
Input: nums = [3,2,4], target = 6
Output: [1,2]

Example 3:
Input: nums = [3,3], target = 6
Output: [0,1]

Constraints:

64.9K 1.6K 2068 Online

12. Given a **non-empty** array of integers `nums`, every element appears *twice* except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

```
class Solution {  
    public int singleNumber(int[] nums) {  
        int result = 0;  
        for (int num : nums) {  
            result ^= num; // XOR all numbers  
        }  
        return result;  
    }  
}
```

OUTPUT:

The screenshot shows a LeetCode problem page for "136. Single Number". The code submitted is the same as above. The "Test Result" section shows the code was accepted with a runtime of 0 ms. It includes three test cases: Case 1 (Input: [2,2,1], Output: 1), Case 2 (Input: [4,1,2,1,2], Output: 4), and Case 3 (Input: [1], Output: 1). The constraints section indicates that the input size is at most 10000 and each element is between -2^31 and 2^31.

13. Given an integer array nums, return true if any value appears **at least twice** in the array, and return false if every element is distinct.

```
class Solution {  
    public boolean containsDuplicate(int[] nums) {  
        HashMap<Integer, Boolean> map = new HashMap<>();  
        for (int num : nums) {  
            if (map.containsKey(num)) {  
                return true;  
            }  
            map.put(num, true);  
        }  
        return false;  
    }  
}
```

OUTPUT:

The screenshot shows the LeetCode platform interface for problem 217. The code editor displays the Java solution provided above. Below the editor, the 'Test Result' section shows the code has been accepted with a runtime of 0 ms, passing three test cases. The input is [1,2,3,1] and the output is true, which matches the expected result for Example 1.

```
class Solution {  
    public boolean containsDuplicate(int[] nums) {  
        HashMap<Integer, Boolean> map = new HashMap<>();  
        for (int num : nums) {  
            if (map.containsKey(num)) {  
                return true;  
            }  
            map.put(num, true);  
        }  
        return false;  
    }  
}
```

14. Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

```
class Solution {
    public void moveZeroes(int[] nums) {
        int insertPos = 0;
        for (int num : nums) {
            if (num != 0) {
                nums[insertPos++] = num;
            }
        }
        while (insertPos < nums.length) {
            nums[insertPos++] = 0;
        }
    }
}
```

OUTPUT :

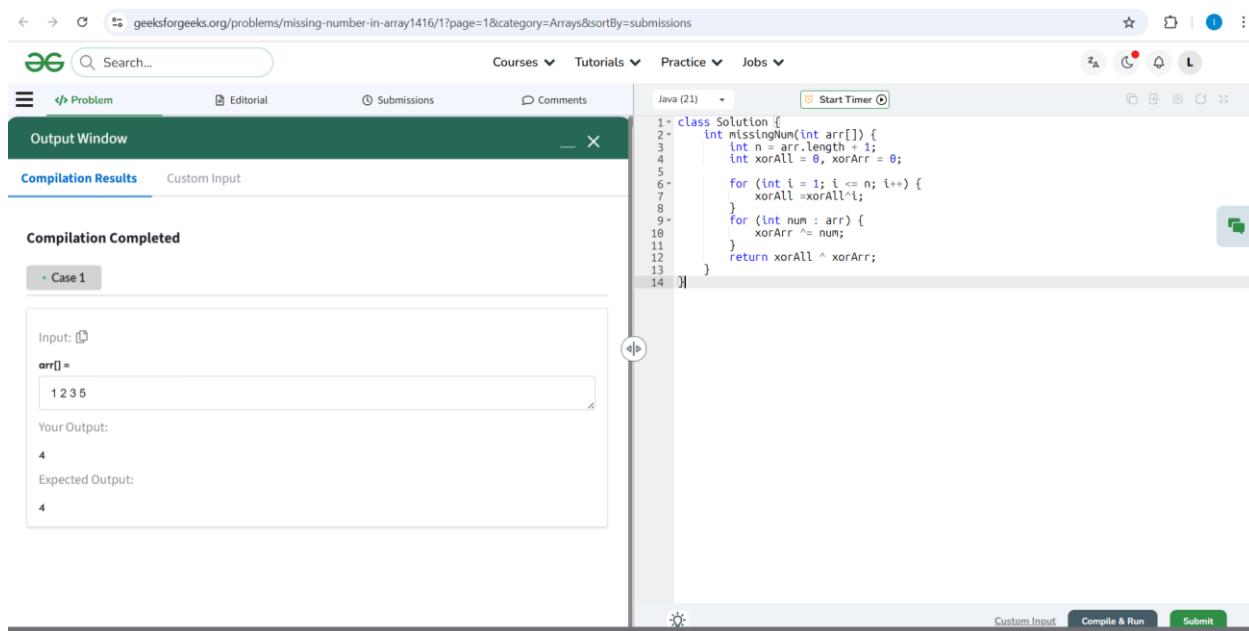
The screenshot shows a Java code submission for the LeetCode problem "Move Zeroes". The code is identical to the one provided above. The submission was successful, with an accepted status and a runtime of 0 ms. The input was [0,1,0,3,12] and the output was [1,3,12,0,0].

```
class Solution {
    public void moveZeroes(int[] nums) {
        int insertPos = 0;
        for (int num : nums) {
            if (num != 0) {
                nums[insertPos++] = num;
            }
        }
        while (insertPos < nums.length) {
            nums[insertPos++] = 0;
        }
    }
}
```

15. You are given an array **arr[]** of size **n - 1** that contains **distinct integers** in the range from 1 to n (inclusive). This array represents a permutation of the integers from 1 to n with **one element missing**. Your task is to identify and return the **missing element**.

```
class Solution {  
    int missingNum(int arr[]) {  
        int n = arr.length + 1;  
        int xorAll = 0, xorArr = 0;  
  
        for (int i = 1; i <= n; i++) {  
            xorAll = xorAll ^ i;  
        }  
        for (int num : arr) {  
            xorArr ^= num;  
        }  
        return xorAll ^ xorArr;  
    }  
}
```

}OUTPUT:



The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is geeksforgeeks.org/problems/missing-number-in-array1416/1?page=1&category=Arrays&sortBy=submissions. The page title is "geeksforgeeks.org/problems/missing-number-in-array1416/1?page=1&category=Arrays&sortBy=submissions". The main content area displays the Java code for the solution. In the "Output Window" section, under "Compilation Results", it shows "Compilation Completed". Under "Case 1", the input is "arr[] = 1 2 3 5" and the output is "4", which matches the expected output.

```
1+ class Solution {  
2+     int missingNum(int arr[]) {  
3+         int n = arr.length + 1;  
4+         int xorAll = 0, xorArr = 0;  
5+  
6+         for (int i = 1; i <= n; i++) {  
7+             xorAll = xorAll ^ i;  
8+         }  
9+         for (int num : arr) {  
10+             xorArr ^= num;  
11+         }  
12+         return xorAll ^ xorArr;  
13+     }  
14+ }
```

16. Given an array of **positive** integers **arr[]**, return the **second largest** element from the array. If the second largest element doesn't exist then return **-1**.

```
class Solution {  
    public int getSecondLargest(int[] arr) {  
        // code here  
        int n=arr.length,k=n-1;  
        Arrays.sort(arr);  
        if(arr.length<2)  
            return -1;  
        while(k-1>=0&&arr[k]==arr[k-1]) {  
            k--;  
        }  
        if(k<=0) return -1;  
        return arr[k-1];  
    }  
}
```

OUTPUT :

The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is geeksforgeeks.org/problems/second-largest3735/1?page=1&category=Arrays&sortBy=submissions. The code area contains the provided Java code. In the Output Window, under Case 1, the input is "arr[] = 12 35 1 10 34 1" and the output is "34". The status bar at the bottom right shows "Custom Input" and "Compile & Run" buttons.

```
1+ class Solution {  
2+     public int getSecondLargest(int[] arr) {  
3+         // code here  
4+         int n=arr.length,k=n-1;  
5+         Arrays.sort(arr);  
6+         if(arr.length<2)  
7+             return -1;  
8+         while(k-1>=0&&arr[k]==arr[k-1]) {  
9+             k--;  
10+        }  
11+        if(k<=0) return -1;  
12+        return arr[k-1];  
13+    }  
14+ }  
15 }
```

17. You are given an array **arr** of positive integers. Your task is to find all the leaders in the array. An element is considered a leader if it is greater than or equal to all elements to its right. The rightmost element is always a leader.

```
class Solution {  
    static ArrayList<Integer> leaders(int arr[]) {  
        // code here  
        ArrayList<Integer> list = new ArrayList<>();  
        int cur = Integer.MIN_VALUE;  
        for (int i = arr.length - 1; i >= 0; i--) {  
            if (arr[i] >= cur) {  
                list.add(arr[i]);  
                cur = arr[i];  
            }  
        }  
        Collections.reverse(list);  
        return list;  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is [geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1?page=1&category=Arrays&sortBy=submissions](https://www.geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1?page=1&category=Arrays&sortBy=submissions). The page title is "Leaders in an array". The main content area displays the Java code for the Solution class. In the "Output Window" section, under "Compilation Results", it shows "Compilation Completed". Under "Case 1", the input is "arr[] = 16 17 4 3 5 2" and the output is "17 5 2", which matches the expected output. At the bottom, there are buttons for "Custom Input", "Compile & Run", and "Submit".

```
1+ class Solution {  
2+     static ArrayList<Integer> leaders(int arr[]) {  
3+         // code here  
4+         ArrayList<Integer> list = new ArrayList<>();  
5+         int cur = Integer.MIN_VALUE;  
6+         for (int i = arr.length - 1; i >= 0; i--) {  
7+             if (arr[i] >= cur) {  
8+                 list.add(arr[i]);  
9+                 cur = arr[i];  
10+            }  
11+        }  
12+        Collections.reverse(list);  
13+        return list;  
14+    }  
15 }
```

18. Given an array **arr[]** of size **n**, containing elements from the range **1** to **n**, and each element appears at most **twice**, return an array of all the integers that appears twice.

You can return the elements in any order but the driver code will print them in sorted order.

```
class Solution {
```

```
    public ArrayList<Integer> findDuplicates(int[] arr) {
```

```
        // code here
```

```
        HashSet<Integer> al=new HashSet<>();
```

```
        HashSet<Integer> a=new HashSet<>();
```

```
        for(int i:arr){
```

```
            if(!al.add(i)){
```

```
                a.add(i);
```

```
            }
```

```
        }
```

```
        return new ArrayList<>(a);
```

```
}
```

```
}
```

OUTPUT :

The screenshot shows the GeeksforGeeks platform interface for solving a problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the problem statement and the user's submitted Java code. The code implements a solution to find duplicates in an array by using two hash sets: one for tracking seen elements and another for collecting duplicates. The code is pasted into a code editor window, and the output window shows the result of running the code with the input [2, 3, 1, 2, 3] and the expected output [2, 3].

```
1- class Solution {
2-     public ArrayList<Integer> findDuplicates(int[] arr) {
3-         // code here
4-         HashSet<Integer> al=new HashSet<>();
5-         HashSet<Integer> a=new HashSet<>();
6-         for(int i:arr){
7-             if(!al.add(i)){
8-                 a.add(i);
9-             }
10-        }
11-        return new ArrayList<>(a);
12-    }
13- }
```

19. Given an array **arr[]** containing only **0s, 1s, and 2s**. Sort the array in ascending order.

Note: You need to solve this problem without utilizing the built-in sort function.

```
class Solution {  
    public static void sort012(int[] arr) {  
        int n = arr.length;  
        int low = 0, mid = 0, high = n - 1;  
        while (mid <= high) {  
            switch (arr[mid]) {  
                case 0:  
                    int temp0 = arr[low];  
                    arr[low] = arr[mid];  
                    arr[mid] = temp0;  
                    low++;  
                    mid++;  
                    break;  
                case 1:  
                    mid++;  
                    break;  
                case 2:  
                    int temp2 = arr[mid];  
                    arr[mid] = arr[high];  
                    arr[high] = temp2;  
                    high--;  
                    break;  
            }  
        }  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area has tabs for Problem, Editorial, Submissions, and Comments. The current tab is 'Problem'. Below the tabs, there's an 'Output Window' tab which is active, showing the 'Compilation Results' section. The code area contains the provided Java solution. The output window shows the input array [0, 1, 2, 1, 0] and the expected output [0, 0, 1, 1, 2, 2]. The status bar at the bottom indicates 'Custom Input' and 'Compile & Run' buttons.

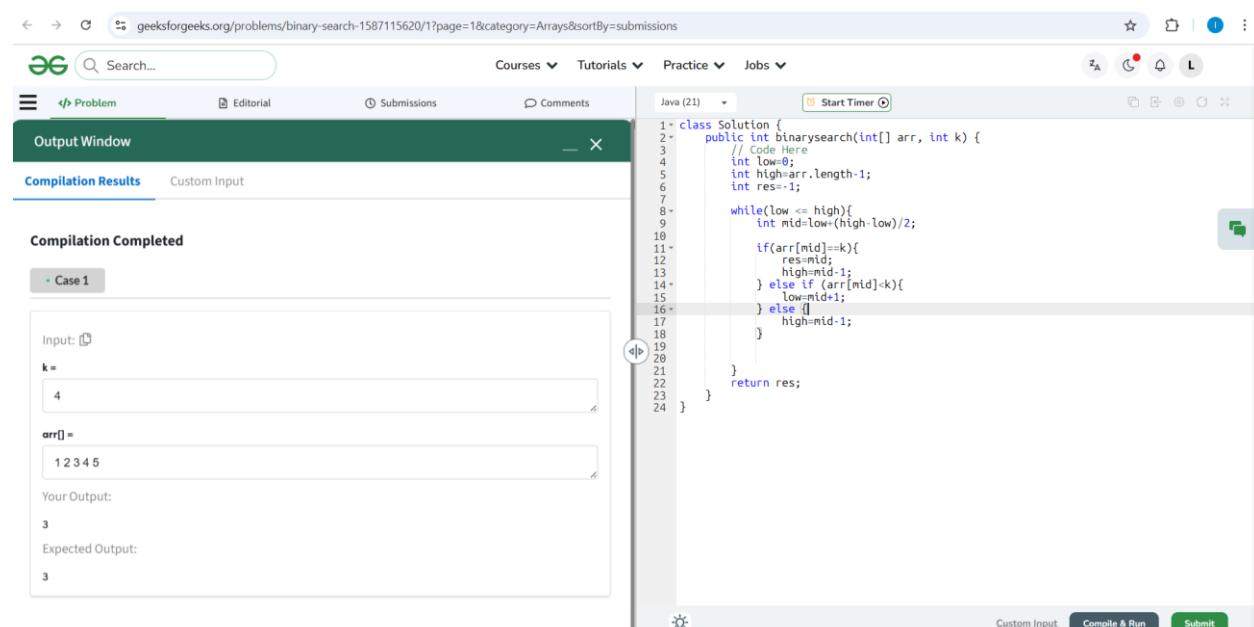
```
1* class Solution {  
2*     public static void sort012(int[] arr) {  
3*         int n = arr.length;  
4*         int low = 0, mid = 0, high = n - 1;  
5*         while (mid <= high) {  
6*             switch (arr[mid]) {  
7*                 case 0:  
8*                     int temp0 = arr[low];  
9*                     arr[low] = arr[mid];  
10*                    arr[mid] = temp0;  
11*                    low++;  
12*                    mid++;  
13*                    break;  
14*                 case 1:  
15*                     mid++;  
16*                     break;  
17*                 case 2:  
18*                     int temp2 = arr[mid];  
19*                     arr[mid] = arr[high];  
20*                     arr[high] = temp2;  
21*                     high--;  
22*                     break;  
23*             }  
24*         }  
25*     }  
26* }
```

20. Given a sorted array **arr[]** and an integer **k**, find the position(0-based indexing) at which **k** is present in the array using binary search. If **k** doesn't exist in **arr[]** return **-1**.

Note: If multiple occurrences are there, please return the smallest index.

```
class Solution {  
    public int binarysearch(int[] arr, int k) {  
        // Code Here  
        int low=0;  
        int high=arr.length-1;  
        int res=-1;  
  
        while(low <= high){  
            int mid=low+(high-low)/2;  
  
            if(arr[mid]==k){  
                res=mid;  
                high=mid-1;  
            } else if (arr[mid]<k){  
                low=mid+1;  
            } else {  
                high=mid-1;  
            }  
        }  
        return res;  
    }  
}
```

OUTPUT:



The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is identical to the one provided above, implementing a binary search algorithm. The editor interface includes tabs for 'Courses', 'Tutorials', 'Practice', and 'Jobs'. Below the code, the 'Output Window' shows the results of a test case. The 'Compilation Results' section indicates 'Compilation Completed'. The 'Case 1' tab is selected, showing the input '4', the array 'arr[] = 1 2 3 4 5', the output '3', and the expected output '3'. The code editor has syntax highlighting for Java keywords and comments.

```
1+ class Solution {  
2+     public int binarysearch(int[] arr, int k) {  
3+         // Code Here  
4+         int low=0;  
5+         int high=arr.length-1;  
6+         int res=-1;  
7+  
8+         while(low <= high){  
9+             int mid=low+(high-low)/2;  
10+            if(arr[mid]==k){  
11+                res=mid;  
12+                high=mid-1;  
13+            } else if (arr[mid]<k){  
14+                low=mid+1;  
15+            } else {  
16+                high=mid-1;  
17+            }  
18+        }  
19+        return res;  
20+    }  
21+ }
```

Programs on Fibonacci Problems

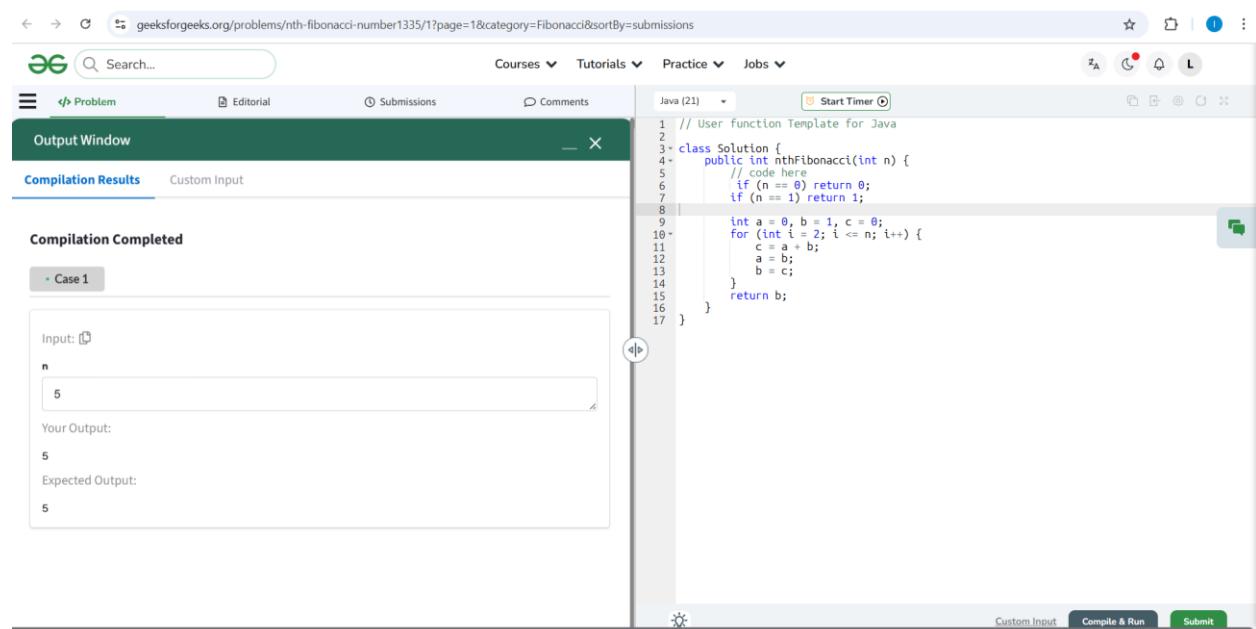
21. Given a non-negative integer **n**, your task is to find the **nth Fibonacci number**.

The [Fibonacci sequence](#) is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1. The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

// User function Template for Java

```
class Solution {  
    public int nthFibonacci(int n) {  
        // code here  
        if (n == 0) return 0;  
        if (n == 1) return 1;  
  
        int a = 0, b = 1, c = 0;  
        for (int i = 2; i <= n; i++) {  
            c = a + b;  
            a = b;  
            b = c;  
        }  
        return b;  
    }  
}
```

OUTPUT :



The screenshot shows a browser window for geeksforgeeks.org/problems/nth-fibonacci-number1335/1?page=1&category=Fibonacci&sortBy=submissions. The page title is "Output Window". The code area contains the provided Java code. The "Compilation Results" section shows "Compilation Completed". The "Case 1" section shows the input "5", the output "5", and the expected output "5". At the bottom, there are buttons for "Custom Input", "Compile & Run", and "Submit".

22. You are given an integer **n**, return the fibonacci series till the **nth**(0-based indexing) term. Since the terms can become very large return the terms **modulo 10^9+7** .

```
class Solution {  
    private static final int MOD = 1_000_000_007;  
    private static final int MAX_N = 100000; // given constraint  
    private static final int[] fib = new int[MAX_N + 1];  
    // Static block to precompute Fibonacci numbers once  
    static {  
        fib[0] = 0;  
        fib[1] = 1;  
        for (int i = 2; i <= MAX_N; i++) {  
            fib[i] = (int)((long) fib[i - 1] + fib[i - 2]) % MOD;  
        }  
    }  
    public static int[] Series(int n) {  
        // Instead of copying, just return a new view up to n  
        int[] res = new int[n + 1];  
        for (int i = 0; i <= n; i++) {  
            res[i] = fib[i];  
        }  
        return res;  
    }  
}
```

OUTPUT:

geeksforgeeks.org/problems/fibonacci-series-up-to-nth-term/1?page=1&category=Fibonacci&sortBy=submissions

Search... Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input

Compilation Completed

Case 1

Input: 5

Your Output:

0 1 1 2 3 5

Expected Output:

0 1 1 2 3 5

Java (21)

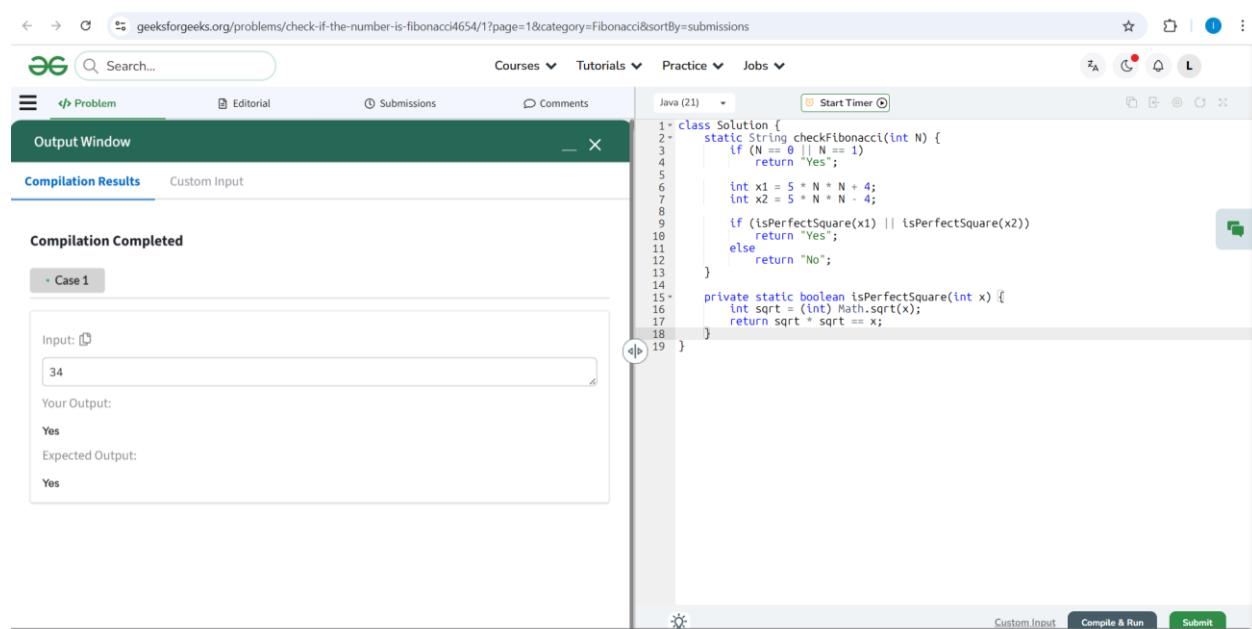
```
1+ class Solution {
2  private static final int MOD = 1_000_000_007;
3  private static final int MAX_N = 100000; // given constraint
4  private static final int[] fib = new int[MAX_N + 1];
5
6  // Static block to precompute Fibonacci numbers once
7  static {
8      fib[0] = 0;
9      fib[1] = 1;
10     for (int i = 2; i <= MAX_N; i++) {
11         fib[i] = (int)((long)fib[i - 1] + fib[i - 2]) % MOD;
12     }
13 }
14
15+ public static int[] Series(int n) {
16     // Instead of copying, just return a new view up to n
17     int[] res = new int[n + 1];
18     for (int i = 0; i <= n; i++) {
19         res[i] = fib[i];
20     }
21     return res;
22 }
23
24 }
```

Custom Input Compile & Run Submit

23. Check if a given number N is the Fibonacci number. A Fibonacci number is a number that occurs in the Fibonacci series.

```
class Solution {  
    static String checkFibonacci(int N) {  
        if (N == 0 || N == 1)  
            return "Yes";  
  
        int x1 = 5 * N * N + 4;  
        int x2 = 5 * N * N - 4;  
  
        if (isPerfectSquare(x1) || isPerfectSquare(x2))  
            return "Yes";  
        else  
            return "No";  
    }  
  
    private static boolean isPerfectSquare(int x) {  
        int sqrt = (int) Math.sqrt(x);  
        return sqrt * sqrt == x;  
    }  
}
```

OUTPUT:



The screenshot shows the GeeksforGeeks IDE interface. The code editor window displays the Java solution for checking if a number is a Fibonacci number. The output window shows the result of running the code with input 34, which outputs "Yes".

```
Output Window  
Compilation Results Custom Input  
Compilation Completed  
Case 1  
Input: 34  
Your Output:  
Yes  
Expected Output:  
Yes
```

```
Java (21) Start Timer  
1+ class Solution {  
2+     static String checkFibonacci(int N) {  
3+         if (N == 0 || N == 1)  
4+             return "Yes";  
5+  
6+         int x1 = 5 * N * N + 4;  
7+         int x2 = 5 * N * N - 4;  
8+  
9+         if (isPerfectSquare(x1) || isPerfectSquare(x2))  
10+            return "Yes";  
11+        else  
12+            return "No";  
13+    }  
14+  
15+    private static boolean isPerfectSquare(int x) {  
16+        int sqrt = (int) Math.sqrt(x);  
17+        return sqrt * sqrt == x;  
18+    }  
19+ }
```

24. The **Fibonacci numbers**, commonly denoted $F(n)$ form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

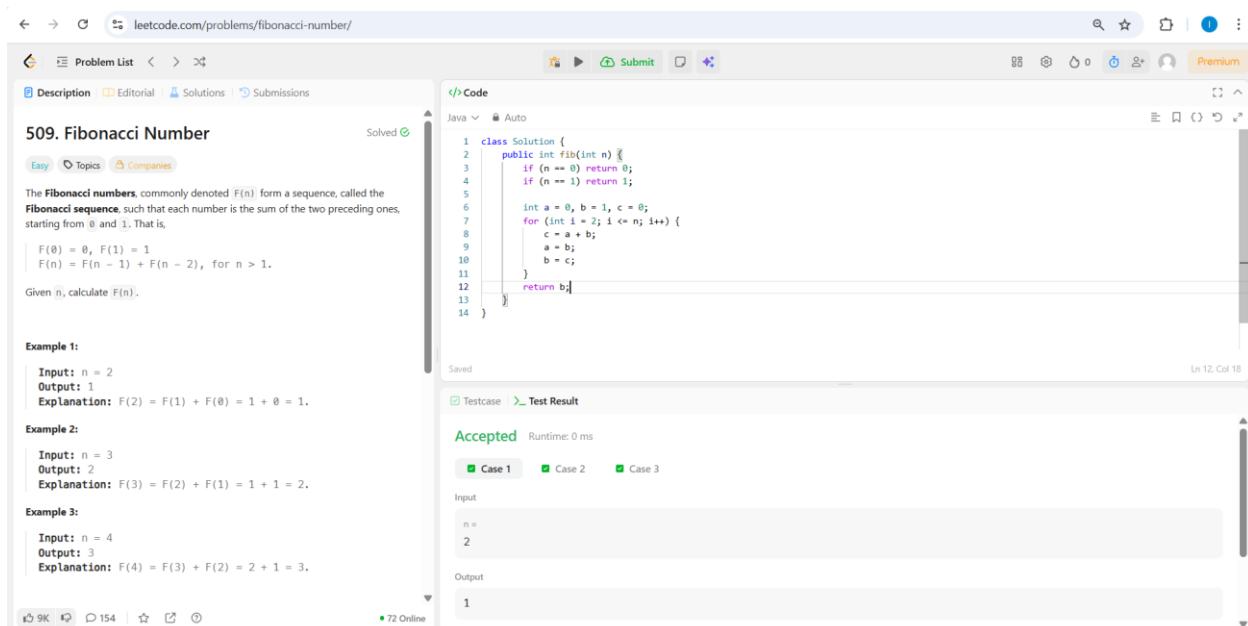
$$F(0) = 0, F(1) = 1$$

$$F(n) = F(n - 1) + F(n - 2), \text{ for } n > 1.$$

Given n , calculate $F(n)$.

```
class Solution {  
    public int fib(int n) {  
        if (n == 0) return 0;  
        if (n == 1) return 1;  
  
        int a = 0, b = 1, c = 0;  
        for (int i = 2; i <= n; i++) {  
            c = a + b;  
            a = b;  
            b = c;  
        }  
        return b;  
    }  
}
```

OUTPUT:



The screenshot shows the LeetCode platform interface for problem 509. The code editor contains the Java implementation of the Fibonacci sequence. The code is accepted, with a runtime of 0 ms. Test cases 1, 2, and 3 are passed. The input is 2 and the output is 1.

```
class Solution {  
    public int fib(int n) {  
        if (n == 0) return 0;  
        if (n == 1) return 1;  
  
        int a = 0, b = 1, c = 0;  
        for (int i = 2; i <= n; i++) {  
            c = a + b;  
            a = b;  
            b = c;  
        }  
        return b;  
    }  
}
```

Programs on String Problems

25. You are given a string **s**. Your task is to determine if the string is a **palindrome**. A string is considered a palindrome if it reads the same forwards and backwards.

```
class Solution {  
    boolean isPalindrome(String s) {  
        int left = 0;  
        int right = s.length() - 1;  
        while (left < right) {  
            if (s.charAt(left) != s.charAt(right)) {  
                return false;  
            }  
            left++;  
            right--;  
        }  
        return true;  
    }  
}
```

OUTPUT:

The screenshot shows a browser window for the GeeksforGeeks website. The URL is [geeksforgeeks.org/problems/palindrome-string0817/1?page=1&category=Strings&sortBy=submissions](https://www.geeksforgeeks.org/problems/palindrome-string0817/1?page=1&category=Strings&sortBy=submissions). The page displays a Java code editor with the provided solution. Below the editor is an 'Output Window' section. Under 'Compilation Results', it says 'Compilation Completed'. In the 'Case 1' input field, the user has entered 'abba'. The 'Your Output:' field shows 'true', and the 'Expected Output:' field also shows 'true'. At the bottom of the window are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1+ class Solution {  
2+     boolean isPalindrome(String s) {  
3+         int left = 0;  
4+         int right = s.length() - 1;  
5+         while (left < right) {  
6+             if (s.charAt(left) != s.charAt(right)) {  
7+                 return false;  
8+             }  
9+             left++;  
10+            right--;  
11+        }  
12+        return true;  
13+    }  
14+ }  
15+  
16+
```

26. You are given a string **s**, and your task is to reverse the string.

```
// User function Template for Java
```

```
class Solution
{
    public static String reverseString(String s)
    {
        return new StringBuilder(s).reverse().toString();
    }
}
```

OUTPUT:

The screenshot shows a problem page from GeeksforGeeks. The title is "Reverse a String". It has a difficulty level of Basic, accuracy of 69.49%, and 433K+ submissions. Points: 1 and Average Time: 15m are also displayed. The problem statement says: "You are given a string **s**, and your task is to reverse the string." Below it, there are examples and constraints.

Examples:

- Input:** s = "Geeks"
Output: "skeeG"
- Input:** s = "for"
Output: "rof"
- Input:** s = "a"
Output: "a"

Constraints:

- $1 \leq s.size() \leq 10^6$
- s contains only alphabetic characters (both uppercase and lowercase).

A "Try more examples" button is available. At the bottom, there are tabs for "Expected Complexities", "Custom Input", "Compile & Run", and "Submit".

27. Given a string **s**, reverse the string without reversing its **individual words**. Words are separated by **dots(.)**.

Note: The string may contain leading or trailing dots(.) or multiple dots(.) between two words. The returned string should only have a single dot(.) separating the words, and **no extra dots** should be included.

```
class Solution {  
    public String reverseWords(String s) {  
        String[] a=s.split("\\.");  
        StringBuilder res=new StringBuilder();  
        for(int i=a.length-1;i>=0;i--){  
            if (a[i].equals("")) continue;  
            if (res.length() > 0) res.append(".");  
            res.append(a[i]);  
        }  
        return res.toString();  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area is titled "Output Window". On the left, there's a "Compilation Results" section showing "Compilation Completed". Below it, the "Input" field contains the string "I.like.this.program.very.much". The "Your Output:" field shows the reversed words: "much.very.program.this.like.i". The "Expected Output:" field also shows the same output. The right side of the interface displays the Java code for the solution. At the bottom, there are buttons for "Custom Input", "Compile & Run", and "Submit".

```
1- class Solution {  
2-     public String reverseWords(String s) {  
3-         String[] a=s.split("\\.");  
4-         StringBuilder res=new StringBuilder();  
5-         for(int i=a.length-1;i>=0;i--){  
6-             if (a[i].equals("")) continue;  
7-             if (res.length() > 0) res.append(".");  
8-             res.append(a[i]);  
9-         }  
10-        return res.toString();  
11-    }  
12- }
```

28. Given a string **s** consisting of **lowercase English Letters**. return the first non-repeating character in **s**. If there is no non-repeating character, return '**\$**'.

```
class Solution {  
    // Function to find the first non-repeating character  
    static char nonRepeatingChar(String s) {  
        int[] freq = new int[26];  
        for (char ch : s.toCharArray()) {  
            freq[ch - 'a']++;  
        }  
        for (char ch : s.toCharArray()) {  
            if (freq[ch - 'a'] == 1) {  
                return ch;  
            }  
        }  
        return '$';  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. In the 'Output Window' section, under 'Compilation Results', it shows 'Compilation Completed'. Below that, 'Case 1' is selected, showing the input 'geeksforgeeks' and the output 'f'. The expected output is also shown as 'f'. At the bottom right of the interface, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1+ class Solution {  
2+     // Function to find the first non-repeating character  
3+     static char nonRepeatingChar(String s) {  
4+         int[] freq = new int[26];  
5+         for (char ch : s.toCharArray()) {  
6+             freq[ch - 'a']++;  
7+         }  
8+         for (char ch : s.toCharArray()) {  
9+             if (freq[ch - 'a'] == 1) {  
10+                 return ch;  
11+             }  
12+         }  
13+     }  
14+     return '$';  
15+ }
```

29. Given two strings **s1** and **s2**. Return true if the string **s2** can be obtained by rotating (**in any direction**) string **s1** by **exactly 2** places, otherwise, false.

Note: Both rotations should be performed in same direction chosen initially.

```
class Solution {  
    public static boolean isRotated(String s1, String s2) {  
        // code here  
        if(s1.length() != s2.length()) {  
            return false;  
        }  
        if(s1.length() <= 2) {  
            return s1.equals(s2);  
        }  
        String left = s1.substring(2) + s1.substring(0, 2);  
        String right = s1.substring(s1.length() - 2) + s1.substring(0, s1.length() - 2);  
        return s2.equals(left) || s2.equals(right);  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. In the 'Output Window' tab, the code is shown with line numbers. The 'Compilation Results' section indicates 'Compilation Completed'. Under 'Case 1', the input values 's1 = amazon' and 's2 = azonam' are displayed. The 'Your Output:' field contains 'true', which matches the 'Expected Output:' field. At the bottom of the window, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1- class Solution {  
2-     public static boolean isRotated(String s1, String s2) {  
3-         // code here  
4-         if(s1.length() != s2.length()) {  
5-             return false;  
6-         }  
7-         if(s1.length() <= 2) {  
8-             return s1.equals(s2);  
9-         }  
10-        String left = s1.substring(2) + s1.substring(0, 2);  
11-        String right = s1.substring(s1.length() - 2) + s1.substring(0, s1.length() - 2);  
12-        return s2.equals(left) || s2.equals(right);  
13-    }  
14-}
```

30. Given a string **s** without spaces, the task is to remove all duplicate characters from it, keeping only the first occurrence.

Note: The original order of characters must be kept the same.

```
import java.util.*;  
class Solution {  
    // method name/signature matches your driver: removeDups(String)  
    public String removeDups(String s) {  
        if (s == null || s.length() == 0) return s;  
        Set<Character> set = new LinkedHashSet<>();  
        for (int i = 0; i < s.length(); i++) {  
            set.add(s.charAt(i));  
        }  
        StringBuilder sb = new StringBuilder(set.size());  
        for (char c : set) sb.append(c);  
        return sb.toString();  
    }  
}  
class Solution1 {  
    public static void main(String[] args) {  
        Solution ob = new Solution();  
        System.out.println(ob.removeDups("zvvo")); // prints "zvo"  
        System.out.println(ob.removeDups("gfg")); // prints "gf"  
        System.out.println(ob.removeDups("aaaa")); // prints "a"  
    }  
}
```

}OUTPUT:

geeksforgeeks.org/problems/remove-duplicates3034/1?page=2&category=Strings&sortBy=submissions

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input

Compilation Completed

Case 1

Input:

s =

Your Output:

zvo

Expected Output:

zvo

Java (21) Start Timer

```
1 import java.util.*;  
2  
3 class Solution {  
4     // method name/signature matches your driver: removeDups(String)  
5     public String removeDups(String s) {  
6         if (s == null || s.length() == 0) return s;  
7         Set<Character> set = new LinkedHashSet<>();  
8         for (int i = 0; i < s.length(); i++) {  
9             set.add(s.charAt(i));  
10        }  
11          
12        StringBuilder sb = new StringBuilder(set.size());  
13        for (char c : set) sb.append(c);  
14        return sb.toString();  
15    }  
16      
17    class Solution1 {  
18        public static void main(String[] args) {  
19            Solution ob = new Solution();  
20            System.out.println(ob.removeDups("zvvo")); // prints "zvo"  
21            System.out.println(ob.removeDups("gfg")); // prints "gf"  
22            System.out.println(ob.removeDups("aaaa")); // prints "a"  
23        }  
24    }  
25      
26}
```

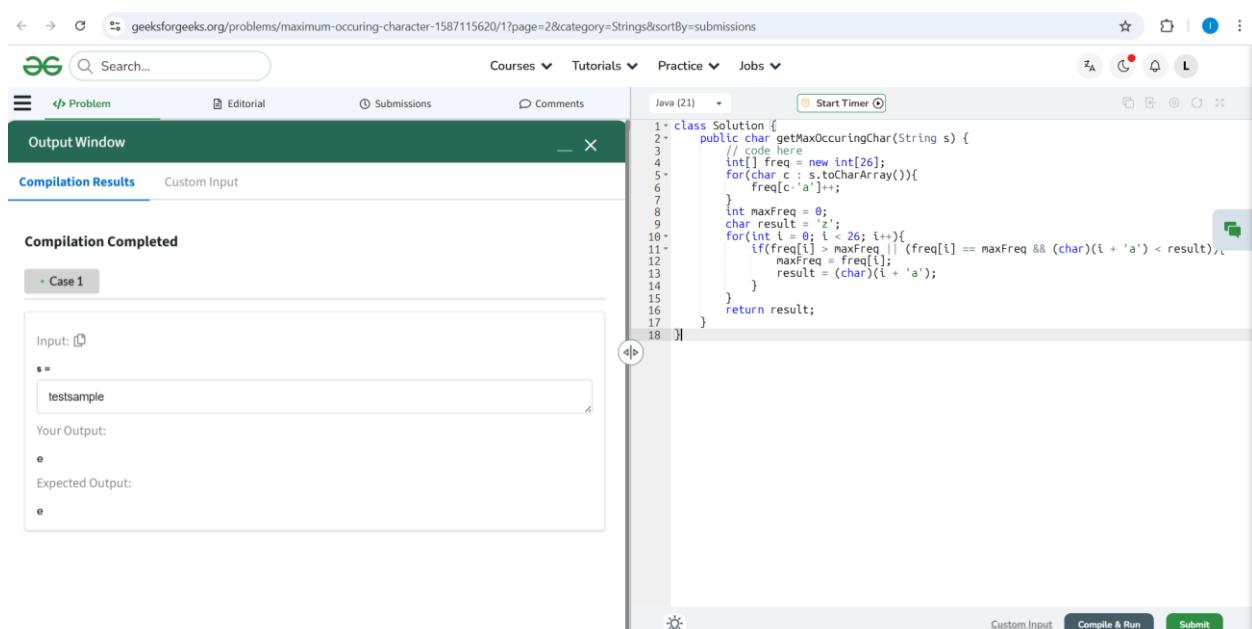
Custom Input Compile & Run Submit

31. Given a string s of lowercase alphabets. The task is to find the maximum occurring character in the string s . If more than one character occurs the maximum number of times then print the lexicographically smaller character.

```
class Solution {
```

```
    public char getMaxOccuringChar(String s) {
        // code here
        int[] freq = new int[26];
        for(char c : s.toCharArray()){
            freq[c-'a']++;
        }
        int maxFreq = 0;
        char result = 'z';
        for(int i = 0; i < 26; i++){
            if(freq[i] > maxFreq || (freq[i] == maxFreq && (char)(i + 'a') < result)){
                maxFreq = freq[i];
                result = (char)(i + 'a');
            }
        }
        return result;
    }
}
```

OUTPUT:



The screenshot shows the GeeksforGeeks online judge interface. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area has tabs for Problem, Editorial, Submissions, and Comments. A search bar is at the top left. The central workspace shows Java code in a code editor window titled "Java (21)". The code implements the logic described in the question. Below the code editor is the "Output Window" which displays the results of a test case. The "Compilation Results" tab is selected, showing "Compilation Completed". Under "Case 1", the "Input" field contains "testsample" and the "Your Output" field also contains "testsample". The "Expected Output" field contains "e". At the bottom right of the workspace are buttons for "Custom Input", "Compile & Run", and "Submit".

32. Given a string **s** consisting of only '0's and '1's, find the last index of the '1' present.

Note: If '1' is not present, return "-1"

```
// User function Template for Java
```

```
class Solution {  
    public int lastIndex(String s) {  
        for(int i=s.length()-1;i>=0;i--){  
            if(s.charAt(i)=='1'){  
                return i;  
            }  
        }  
        return -1;  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. On the left, the 'Output Window' shows the compilation results, indicating 'Compilation Completed'. The 'Input' field contains '00001', and the 'Your Output' field shows '4', which matches the 'Expected Output'. At the bottom, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1 // User function Template for Java  
2  
3 class Solution {  
4     public int lastIndex(String s) {  
5         for(int i=s.length()-1;i>=0;i--){  
6             if(s.charAt(i)=='1'){  
7                 return i;  
8             }  
9         }  
10        return -1;  
11    }  
12 }
```

33. Given two non-empty strings **s1** and **s2**, consisting only of lowercase English letters, determine whether they are **anagrams** of each other or not.

Two strings are considered **anagrams** if they contain the **same characters** with exactly the **same frequencies**, regardless of their order.

```
class Solution {  
    public static boolean areAnagrams(String s1, String s2) {  
        char c1[] = s1.toCharArray();  
        char c2[] = s2.toCharArray();  
        Arrays.sort(c1);  
        Arrays.sort(c2);  
        return Arrays.equals(c1,c2);  
    }  
}
```

OUTPUT:

The screenshot shows a Java IDE interface on the GeeksforGeeks website. The code editor contains the provided Java solution. The output window shows the compilation completed successfully. The input fields show `s1 = geeks` and `s2 = kseeg`. The output field shows `true`, which matches the expected output of `true`.

```
1 class Solution {  
2     public static boolean areAnagrams(String s1, String s2) {  
3         char c1[] = s1.toCharArray();  
4         char c2[] = s2.toCharArray();  
5         Arrays.sort(c1);  
6         Arrays.sort(c2);  
7         return Arrays.equals(c1,c2);  
8     }  
9 }
```

Compilation Completed

Case 1

Input:
s1 =
geeks
s2 =
kseeg

Your Output:
true

Expected Output:
true

34. You are given a string s , consisting of lowercase alphabets. Your task is to remove consecutive duplicate characters from the string.

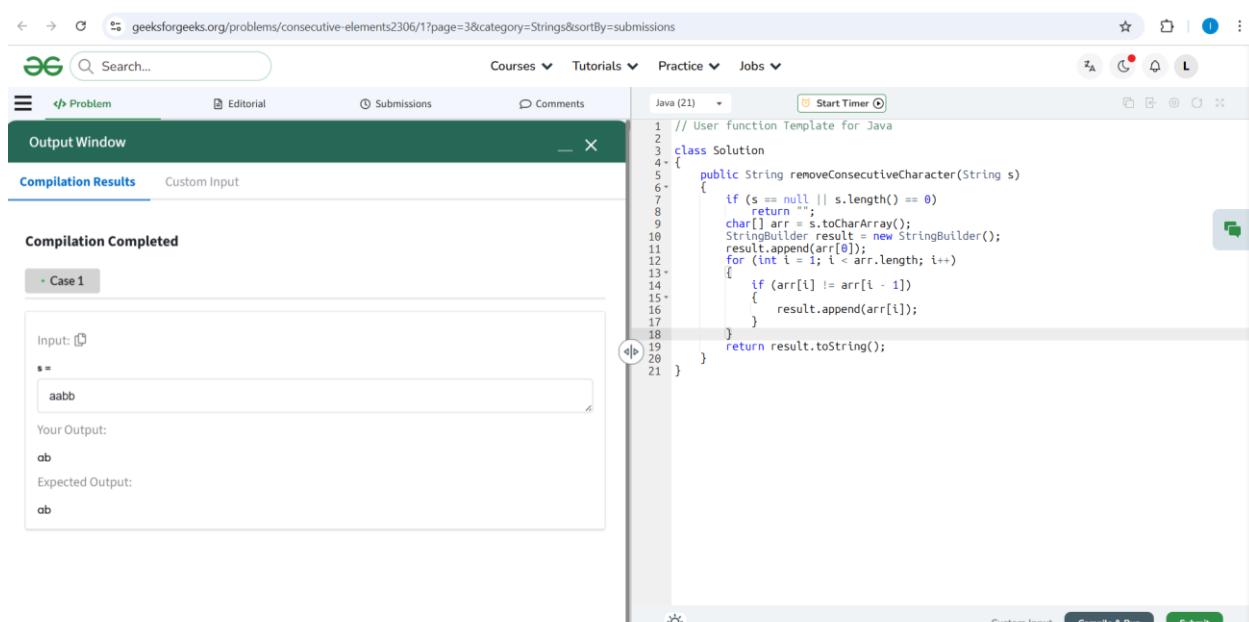
```
// User function Template for Java

class Solution

{
    public String removeConsecutiveCharacter(String s)
    {
        if(s == null || s.length() == 0)
            return "";

        char[] arr = s.toCharArray();
        StringBuilder result = new StringBuilder();
        result.append(arr[0]);
        for (int i = 1; i < arr.length; i++)
        {
            if (arr[i] != arr[i - 1])
            {
                result.append(arr[i]);
            }
        }
        return result.toString();
    }
}
```

OUTPUT:



The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the solution. Below the code, the 'Output Window' section shows the results of a test case. The 'Compilation Results' tab is selected, displaying 'Compilation Completed'. The 'Case 1' tab is active, showing the input 'aabb', the output 'ab', and the expected output 'ab'. At the bottom of the window are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1 // User function Template for Java
2
3 class Solution
4 {
5     public String removeConsecutiveCharacter(String s)
6     {
7         if(s == null || s.length() == 0)
8             return "";
9         char[] arr = s.toCharArray();
10        StringBuilder result = new StringBuilder();
11        result.append(arr[0]);
12        for (int i = 1; i < arr.length; i++)
13        {
14            if (arr[i] != arr[i - 1])
15            {
16                result.append(arr[i]);
17            }
18        }
19    }
20
21 }
```

Programs on Series Problems

35. Juggler Sequence is a series of integers in which the first term starts with a positive integer number a and the remaining terms are generated from the immediate previous term using the below recurrence relation:

$$a_{k+1} = \begin{cases} \lfloor a_k^{1/2} \rfloor & \text{for even } a_k \\ \lfloor a_k^{3/2} \rfloor & \text{for odd } a_k, \end{cases}$$

Given a number n, find the Juggler Sequence for this number as the first term of the sequence until it becomes 1.

```
import java.util.*;
class Solution {
    public ArrayList<Long> jugglerSequence(long n) {
        ArrayList<Long> seq = new ArrayList<>();
        seq.add(n);
        while (n != 1) {
            if (n % 2 == 0) {
                n = (long) Math.floor(Math.sqrt(n));
            } else {
                n = (long) Math.floor(Math.pow(n, 1.5));
            }
            seq.add(n);
        }
        return seq;
    }
}
```

OUTPUT:

The screenshot shows a web-based programming environment on the GeeksforGeeks website. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area is titled "Practice Coding Problems" under "National Skillup- Free Courses". A "Problem of the Day" section displays the following Java code:

```
jugglerSequence(long n) {  
    List<Long> seq = new ArrayList<>();  
    if (n % 2 == 0) {  
        n = (long) Math.floor(Math.sqrt(n));  
    } else {  
        n = (long) Math.floor(Math.pow(n, 1.5));  
    }  
    seq.add(n);  
    return seq;  
}
```

The "Output Window" on the left shows the following interaction:

- Input: `n = 9`
- Your Output: `9 27 140 11 36 6 2 1`
- Expected Output: `9 27 140 11 36 6 2 1`

At the bottom right of the interface are buttons for "Custom Input", "Compile & Run", and "Submit".

36. Given a series 9, 33, 73, 129... Find the n-th term of the series.

class Solution

```
{  
    static long nthOfSeries(long n)  
    {  
        return 8 * n *n+1;  
    }  
}
```

OUTPUT:

The screenshot shows a Java IDE interface on the GeeksforGeeks website. The code editor contains the provided Java code. The output window shows the result of running the code with input 4, which produced the output 129. The expected output is also shown as 129.

```
Output Window  
Compilation Results Custom Input  
Compilation Completed  
Case 1  
Input: 4  
Your Output:  
129  
Expected Output:  
129
```

```
Java (21) Start Timer  
1 class Solution  
2 {  
3     static long nthOfSeries(long n)  
4     {  
5         return 8 * n *n+1;  
6     }  
7 }
```

37. Given a series of numbers 3,10,21,36, and series starting from N = 1, find the pattern and output the N'th value of the above series.

```
class Solution
{
    static int differenceSeries(int N)
    {
        return N * (2 *N+1);
    }
}
```

OUTPUT:

The screenshot shows a web browser window for the GeeksforGeeks website. The URL is [geeksforgeeks.org/problems/difference-series4345/1?page=1&category=series&sortBy=submissions](https://www.geeksforgeeks.org/problems/difference-series4345/1?page=1&category=series&sortBy=submissions). The page displays a Java code editor with the following code:

```
class Solution
{
    static int differenceSeries(int N)
    {
        return N * (2 *N+1);
    }
}
```

The code is submitted and the status is "Compilation Completed". The input field contains "1" and the output field also contains "1", indicating a successful compilation and execution of the code.

38. Given a number x , your task is to find if this number is Deficient number or not. A number x is said to be Deficient Number if sum of all the divisors of the number denoted by $\text{divisorsSum}(x)$ is less than twice the value of the number x . And the difference between these two values is called the **deficiency**.

Mathematically, if below condition holds the number is said to be Deficient:

$$\text{divisorsSum}(x) < 2*x$$

$$\text{deficiency} = (2*x) - \text{divisorsSum}(x)$$

```
class Solution
```

```
{
```

```
    static String isDeficient(long x)
```

```
{
```

```
    long sum = 0;
```

```
    for (long i = 1; i * i <= x; i++)
```

```
{
```

```
    if (x % i == 0)
```

```
{
```

```
        sum += i;
```

```
        long j = x / i;
```

```
        if (j != i) sum += j;
```

```
}
```

```
}
```

```
    return (sum < 2L * x) ? "YES" : "NO";
```

```
}
```

```
}
```

OUTPUT:

The screenshot shows a Java code editor and an output window for a programming challenge on GeeksforGeeks.

Java Code:

```
1 class Solution
2 {
3     static String isDeficient(long x)
4     {
5         long sum = 0;
6         for (long i = 1; i * i <= x; i++)
7         {
8             if (x % i == 0)
9             {
10                 sum += i;
11                 long j = x / i;
12                 if (j != i) sum += j;
13             }
14         }
15     return (sum < 2L * x) ? "YES" : "NO";
16 }
17 }
```

Output Window:

Compilation Results (Case 1)

Input: 21

Your Output: YES

Expected Output: YES

Buttons at the bottom: Custom Input, Compile & Run, Submit

Programs on Prime Numbers Problems

39. Given a number **n**, determine whether it is a **prime number** or not.

Note: A prime number is a number greater than 1 that has no positive divisors other than 1 and itself.

```
class Solution {  
    static boolean isPrime(int n) {  
        if (n <= 1) {  
            return false;  
        }  
        for (int i = 2; i <= Math.sqrt(n); i++) {  
            if (n % i == 0) {  
                return false;  
            }  
        }  
        return true;  
    }  
}
```

OUTPUT:

The screenshot shows a Java IDE interface on geeksforgeeks.org. The code in the editor is:

```
1+ class Solution {  
2+     static boolean isPrime(int n) {  
3+         if (n <= 1) {  
4+             return false;  
5+         }  
6+         for (int i = 2; i <= Math.sqrt(n); i++) {  
7+             if (n % i == 0) {  
8+                 return false;  
9+             }  
10+        }  
11+        return true;  
12+    }  
13 }
```

The output window shows the following results for input 7:

- Compilation Completed
- Case 1
- Input: 7
- Your Output: true
- Expected Output: true

40. Given a number **n**, your task is to find the largest prime factor of n.

```
class Solution {  
    static long largestPrimeFactor(long n) {  
        long maxPrime = -1;  
        while (n % 2 == 0) {  
            maxPrime = 2;  
            n /= 2;  
        }  
        for (long i = 3; i * i <= n; i += 2) {  
            while (n % i == 0) {  
                maxPrime = i;  
                n /= i;  
            }  
        }  
        if (n > 2) {  
            maxPrime = n;  
        }  
  
        return maxPrime;  
    }  
}
```

OUTPUT:

The screenshot shows a web-based programming environment on geeksforgeeks.org. The URL in the address bar is geeksforgeeks.org/problems/largest-prime-factor2601/1?page=1&category=Prime%20Number&sortBy=submissions. The page title is "Output Window". The main content area displays a Java code editor with the following code:

```
Java (21) Start Timer
1 class Solution {
2     static long largestPrimeFactor(long n) {
3         long maxPrime = -1;
4         while (n % 2 == 0) {
5             maxPrime = 2;
6             n /= 2;
7         }
8         for (long i = 3; i * i <= n; i += 2) {
9             while (n % i == 0) {
10                 maxPrime = i;
11                 n /= i;
12             }
13         }
14         if (n > 2) {
15             maxPrime = n;
16         }
17     }
18     return maxPrime;
19 }
20 }
```

The code is a Java function named `largestPrimeFactor` that takes a long integer `n` as input and returns the largest prime factor of `n`. The code uses a combination of a while loop for even numbers and a for loop for odd numbers starting from 3. It also handles the case where `n` is greater than 2.

On the left side, there is an "Output Window" section with tabs for "Compilation Results" and "Custom Input". Under "Compilation Completed", it shows "Case 1" with the following details:

- Input: `n = 5`
- Your Output: `5`
- Expected Output: `5`

At the bottom right, there are buttons for "Custom Input", "Compile & Run", and "Submit".

41. Given a number **n**. Find its **unique** prime factors in **increasing order**.

```
import java.util.ArrayList;
class Solution {
    public static ArrayList<Integer> primeFac(int n) {
        ArrayList<Integer> primes = new ArrayList<>();
        if (n % 2 == 0) {
            if (!primes.contains(2)) {
                primes.add(2);
            }
        }
        while (n % 2 == 0) {
            n /= 2;
        }
        for (int i = 3; i * i <= n; i += 2) {
            if (n % i == 0) {
                if (!primes.contains(i)) {
                    primes.add(i);
                }
            }
            while (n % i == 0) {
                n /= i;
            }
        }
        if (n > 2) {
            if (!primes.contains(n)) {
                primes.add(n);
            }
        }
    }
    return primes;
}
```

OUTPUT:

geeksforgeeks.org/problems/prime-factors5052/1?page=1&category=Prime%20Number&sortBy=submissions

Search... Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window X

Compilation Results Custom Input

Compilation Completed

Case 1

Input: n = 100

Your Output: [2, 5]

Expected Output: [2, 5]

Java (21) Start Timer

```
1- import java.util.ArrayList;
2- class Solution {
3-     public static ArrayList<Integer> primeFac(int n) {
4-         ArrayList<Integer> primes = new ArrayList<>();
5-         if (n % 2 == 0) {
6-             if (!primes.contains(2)) {
7-                 primes.add(2);
8-             }
9-             while (n % 2 == 0) {
10-                 n /= 2;
11-             }
12-         }
13-         for (int i = 3; i * i <= n; i += 2) {
14-             if (n % i == 0) {
15-                 if (!primes.contains(i)) {
16-                     primes.add(i);
17-                 }
18-                 while (n % i == 0) {
19-                     n /= i;
20-                 }
21-             }
22-         }
23-         if (n > 2) {
24-             if (!primes.contains(n)) {
25-                 primes.add(n);
26-             }
27-         }
28-     }
29-     return primes;
30- }
31- }
```

Custom Input Compile & Run Submit

42. Given a number n, find out if n can be expressed as a+b, where both a and b are prime numbers. If such a pair exists, return the values of a and b, otherwise return [-1,-1] as an array of size 2.

Note: If [a, b] is one solution with a \leq b, and [c, d] is another solution with c \leq d, and a < c then [a, b] is considered as our answer.

```
class Solution {  
    public static ArrayList<Integer> getPrimes(int n)  
    {  
        ArrayList<Integer> res = new ArrayList<>();  
        for (int i=2;i<=n/2;i++) {  
            if (isPrime(i) && isPrime(n-i))  
            {  
                res.add(i);  
                res.add(n-i);  
                return res;  
            }  
        }  
        res.add(-1);  
        res.add(-1);  
        return res;  
    }  
    private static boolean isPrime(int num)  
    {  
        if (num<2)  
            return false;  
        for (int i=2; i*i<=num; i++) {  
            if (num % i == 0)  
                return false;  
        }  
        return true;  
    }  
}
```

OUTPUT:

The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is identical to the one above, with line numbers 1 through 28. The editor has tabs for 'Java (21)', 'Start Timer', and 'Run'. Below the code, there's an 'Output Window' tab. Under 'Compilation Results', it says 'Compilation Completed'. Under 'Case 1', the 'Input' field contains '8', and the 'Your Output' field shows '3 5'. The 'Expected Output' field also shows '3 5'. At the bottom right, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

```
1+ class Solution {  
2+     public static ArrayList<Integer> getPrimes(int n)  
3+     {  
4+         ArrayList<Integer> res = new ArrayList<>();  
5+         for (int i=2;i<=n/2;i++) {  
6+             if (isPrime(i) && isPrime(n-i))  
7+             {  
8+                 res.add(i);  
9+                 res.add(n-i);  
10+                return res;  
11+            }  
12+        }  
13+        res.add(-1);  
14+        res.add(-1);  
15+        return res;  
16+    }  
17+    private static boolean isPrime(int num)  
18+    {  
19+        if (num<2)  
20+            return false;  
21+        for (int i=2; i*i<=num; i++) {  
22+            if (num % i == 0)  
23+                return false;  
24+        }  
25+        return true;  
26+    }  
27+}  
28+
```

43. Given two integers M and N, generate all primes between M and N including M and N.

```
import java.util.ArrayList;
class Solution {
    ArrayList<Integer> primeRange(int M, int N) {
        ArrayList<Integer> result = new ArrayList<>();
        if (N < 2) return result;
        int limit = (int) Math.sqrt(N);
        boolean[] isPrimeSmall = new boolean[limit + 1];
        for (int i = 2; i <= limit; i++) {
            isPrimeSmall[i] = true;
        }
        for (int i = 2; i * i <= limit; i++) {
            if (isPrimeSmall[i]) {
                for (int j = i * i; j <= limit; j += i) {
                    isPrimeSmall[j] = false;
                }
            }
        }
        ArrayList<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= limit; i++) {
            if (isPrimeSmall[i]) {
                primes.add(i);
            }
        }
        boolean[] isPrimeRange = new boolean[N - M + 1];
        for (int i = 0; i < isPrimeRange.length; i++) {
            isPrimeRange[i] = true;
        }
        for (int prime : primes) {
            int start = Math.max(prime * prime, ((M + prime - 1) / prime) * prime);
            for (int j = start; j <= N; j += prime) {
                isPrimeRange[j - M] = false;
            }
        }
    }
}
```

```

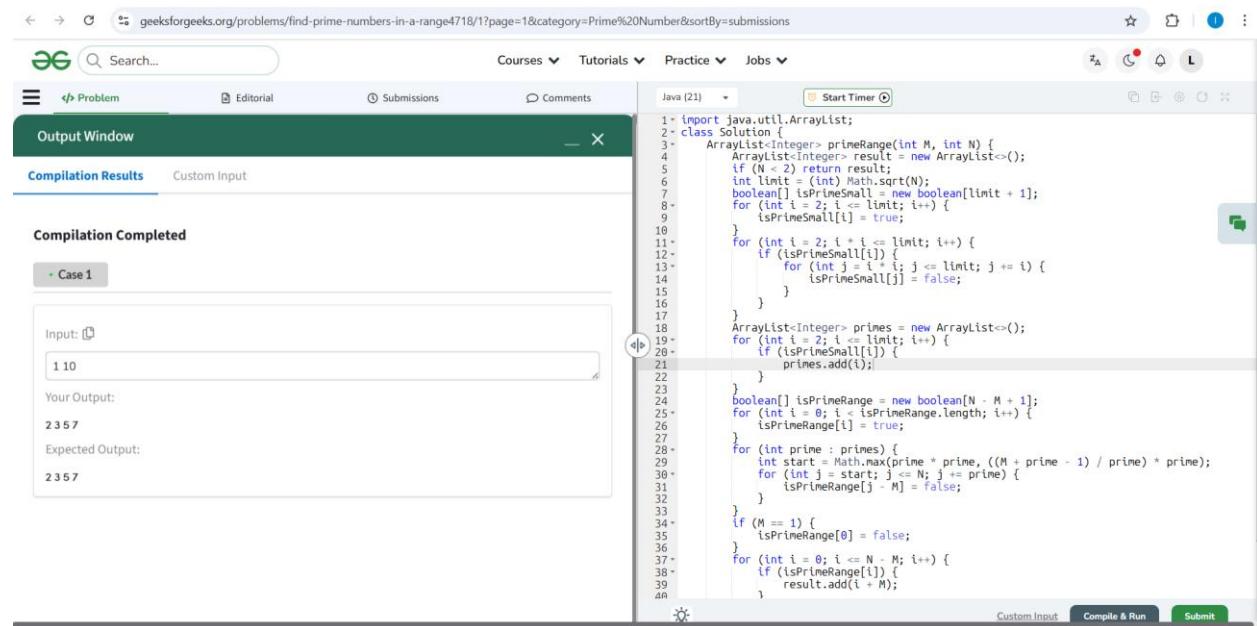
if (M == 1) {
    isPrimeRange[0] = false;
}

for (int i = 0; i <= N - M; i++) {
    if (isPrimeRange[i]) {
        result.add(i + M);
    }
}

return result;
}
}

```

OUTPUT:



The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is for finding prime numbers in a given range [M, N]. It uses a sieve-like approach where it marks non-prime numbers in the isPrimeSmall array. The output window shows the input 1 10, the output 2 3 5 7, and the expected output 2 3 5 7.

```

1 import java.util.ArrayList;
2 class Solution {
3     ArrayList<Integer> primeRange(int M, int N) {
4         ArrayList<Integer> result = new ArrayList<>();
5         if (M > N) return result;
6         int limit = (int) Math.sqrt(N);
7         boolean[] isPrimeSmall = new boolean[limit + 1];
8         for (int i = 2; i <= limit; i++) {
9             isPrimeSmall[i] = true;
10        }
11        for (int i = 2; i * i <= limit; i++) {
12            if (!isPrimeSmall[i]) {
13                for (int j = i + i; j <= limit; j += i) {
14                    isPrimeSmall[j] = false;
15                }
16            }
17        }
18        ArrayList<Integer> primes = new ArrayList<>();
19        for (int i = 2; i <= limit; i++) {
20            if (!isPrimeSmall[i]) {
21                primes.add(i);
22            }
23        }
24        boolean[] isPrimeRange = new boolean[N - M + 1];
25        for (int i = 0; i < isPrimeRange.length; i++) {
26            isPrimeRange[i] = true;
27        }
28        for (int prime : primes) {
29            int start = Math.max(prime * prime, ((M + prime - 1) / prime) * prime);
30            for (int j = start; j <= N; j += prime) {
31                isPrimeRange[j - M] = false;
32            }
33        }
34        if (M == 1) {
35            isPrimeRange[0] = false;
36        }
37        for (int i = 0; i <= N - M; i++) {
38            if (isPrimeRange[i]) {
39                result.add(i + M);
40            }
41        }
42    }
43 }

```

44. Given an array of **n** integers. Find the **minimum** non-negative number to be inserted in array, so that sum of all elements of array becomes **prime**.

```
class Solution {  
    static boolean isPrime(int num) {  
        if (num <= 1) return false;  
        if (num == 2) return true;  
        if (num % 2 == 0) return false;  
        for (int i = 3; i * i <= num; i += 2) {  
            if (num % i == 0) return false;  
        }  
        return true;  
    }  
    public int minNumber(int[] arr, int n) {  
        int sum = 0;  
        for (int num : arr) {  
            sum += num;  
        }  
        int toAdd = 0;  
        while (!isPrime(sum + toAdd)) {  
            toAdd++;  
        }  
        return toAdd;  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area displays the Java code for the Solution class. The code defines a static boolean method isPrime that checks if a number is prime by testing divisibility from 3 up to the square root of the number. It also defines a public int method minNumber that calculates the minimum non-negative integer to add to an array's sum to make it prime. The code is pasted into a code editor window.

Output Window

Compilation Results Custom Input

Compilation Completed

Case 1

Input: 5
2 4 6 8 12

Your Output: 5

Expected Output: 5

Java [21]

```
1 - class Solution {  
2 -     static boolean isPrime(int num) {  
3 -         if (num <= 1) return false;  
4 -         if (num == 2) return true;  
5 -         if (num % 2 == 0) return false;  
6 -         for (int i = 3; i * i <= num; i += 2) {  
7 -             if (num % i == 0) return false;  
8 -         }  
9 -         return true;  
10-    }  
11-    public int minNumber(int[] arr, int n) {  
12-        int sum = 0;  
13-        for (int num : arr) {  
14-            sum += num;  
15-        }  
16-        int toAdd = 0;  
17-        while (!isPrime(sum + toAdd)) {  
18-            toAdd++;  
19-        }  
20-        return toAdd;  
21-    }  
22-}
```

Custom Input Compile & Run Submit

Programs on Pattern Problems

45. Given a number **n**, print a **sequence of numbers** starting from **n**. Each next number in the sequence is **n - 5**, and this continues recursively until the number becomes **less than or equal to 0**. After that, print the sequence in reverse order, **adding 5 each time**, until it reaches back to the original number **n**.

Note: You must not use loops.

class Solution {

```
static void seq(int n,ArrayList<Integer> al) {
    al.add(n);
    if(n<=0)
        return;
    seq(n-5,al);
    al.add(n);
}

public ArrayList<Integer> pattern(int n) {
    ArrayList<Integer> al=new ArrayList<>();
    seq(n,al);
    return al;
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving a programming problem. The URL in the address bar is <https://www.geeksforgeeks.org/problems/print-pattern3549/1?page=1&category=pattern-printing&difficulty=Easy&status=solved&sortBy=submissions>. The page title is "Problem". The main content area displays the Java code for the solution. On the left, there's an "Output Window" section showing the input "n = -16" and the expected output "[-16]". The right side shows the code editor with the Java code and a "Start Timer" button. At the bottom, there are buttons for "Custom Input", "Compile & Run", and "Submit".

```
1- class Solution {
2-     static void seq(int n,ArrayList<Integer> al) {
3-         al.add(n);
4-         if(n<=0)
5-             return;
6-         seq(n-5,al);
7-         al.add(n);
8-     }
9-     public ArrayList<Integer> pattern(int n) {
10-         ArrayList<Integer> al=new ArrayList<>();
11-         seq(n,al);
12-         return al;
13-     }
14- }
```

46. Given an integer **n**, print an inverted isosceles triangle of stars such that the height of the triangle is **n**.

```
class Solution {  
    static String[] invIsoTriangle(int N) {  
        // code here  
        String[] Y = new String[N];  
        char[] stars = new char[2 * N - 1];  
        Arrays.fill(stars, '*');  
        for (int i = 0; i < N; i++)  
        {  
            Y[i] = String.valueOf(stars);  
            stars[i] = ' ';  
            stars[stars.length - i - 1] = ' ';  
        }  
        return Y;  
    }  
};
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is <https://www.geeksforgeeks.org/problems/inverted-triangle-of-stars0110/1?page=1&category=pattern-printing&difficulty=Easy&status=solved&sortBy=submissions>. The page title is "Inverted Triangle of Stars". The main content area has tabs for "Courses", "Tutorials", "Practice", and "Jobs". Below these is a search bar and a navigation menu with "Problem", "Editorial", "Submissions", and "Comments". A "Start Timer" button is visible. The code editor on the right contains the provided Java code. The output window on the left shows the input "n = 4" and the expected output "*****\n****\n***\n*\n". The status bar at the bottom indicates "Compilation Completed" and "Case 1".

```
1- class Solution {  
2-     static String[] invIsoTriangle(int N) {  
3-         // code here  
4-         String[] Y = new String[N];  
5-         char[] stars = new char[2 * N - 1];  
6-         Arrays.fill(stars, '*');  
7-         for (int i = 0; i < N; i++)  
8-         {  
9-             Y[i] = String.valueOf(stars);  
10-            stars[i] = ' ';  
11-            stars[stars.length - i - 1] = ' ';  
12-        }  
13-        return Y;  
14-    }  
15-};
```

Programs on Matrices Problems

47. Given a **row-wise sorted** matrix **mat[][]** of size **n*m**, where the number of rows and columns is always **odd**. Return the **median** of the matrix.

```
import java.util.*;

class Solution {

    // Function to count elements <= target in a sorted row using binary search
    private static int countSmallerEqual(int[] row, int target) {
        int l = 0, r = row.length - 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            if (row[mid] <= target) {
                l = mid + 1; // move right
            } else {
                r = mid - 1; // move left
            }
        }
        return l; // number of elements <= target
    }

    public int median(int[][] matrix) {
        int r = matrix.length;
        int c = matrix[0].length;

        int low = 1, high = 2000; // given constraints
        int desired = (r * c) / 2;

        while (low <= high) {
            int mid = (low + high) / 2;

            // count how many elements <= mid
            int count = 0;
            for (int i = 0; i < r; i++) {
                count += countSmallerEqual(matrix[i], mid);
            }

            if (count <= desired) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return low;
    }
}
```

```
}
```

OUTPUT:

The screenshot shows a Java code editor on the GeeksforGeeks platform. The code is a solution for finding the median in a row-wise sorted matrix. It uses a binary search approach on the rows to find the median element.

```
1+ import java.util.*;
2+
3+ class Solution {
4+     private static int countSmallerEqual(int[] row, int target) {
5+         int l = 0, r = row.length - 1;
6+         while (l <= r) {
7+             int mid = (l + r) / 2;
8+             if (row[mid] <= target) {
9+                 l = mid + 1;
10+            } else {
11+                r = mid - 1;
12+            }
13+        }
14+        return l;
15+    }
16+
17+    public int median(int[][] matrix) {
18+        int r = matrix.length;
19+        int c = matrix[0].length;
20+
21+        int low = 1, high = 2000;
22+        int desired = (r * c) / 2;
23+
24+        while (low <= high) {
25+            int mid = (low + high) / 2;
26+
27+            int count = 0;
28+            for (int i = 0; i < r; i++) {
29+                count += countSmallerEqual(matrix[i], mid);
30+            }
31+
32+            if (count <= desired) {
33+                low = mid + 1;
34+            } else {
35+                high = mid - 1;
36+            }
37+
38+        }
39+        return low;
40+    }
}
```

The input fields show the following values:

- Input: `3`
- `n = 3`
- `m = 3`
- `mat[][] =`
`1 3 5`
`2 6 9`
`3 8 9`

The output field shows `5`, which is the expected output.

48. You are given a square matrix of size $n \times n$. Your task is to find the **transpose** of the given matrix.

The **transpose** of a matrix is obtained by converting all the rows to columns and all the columns to rows.

```
class Solution {  
    public ArrayList<ArrayList<Integer>> transpose(int[][] mat) {  
        // code here  
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();  
        for(int i = 0; i < mat.length; i++) {  
            ArrayList<Integer> temp = new ArrayList<>();  
            for(int j = 0; j < mat[0].length; j++) {  
                temp.add(mat[j][i]);  
            }  
            list.add(temp);  
        }  
        return list;  
    }  
}
```

OUTPUT:

The screenshot shows the GeeksforGeeks platform interface for solving the transpose problem. The code editor contains the provided Java solution. In the 'Output Window' tab, under 'Compilation Results', it shows 'Compilation Completed'. The 'Input' field has 'n = 4' and 'mat[][] = [[1, 1, 1], [2, 2, 2]]'. The 'Your Output:' field displays the expected output: [[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]. The 'Expected Output:' field also shows this same list. At the bottom, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

49. Given a 2D **binary matrix**(1-based indexed) **mat** of dimensions **nxm** , determine the **row** that contains the **minimum number of 1's**.

Note: The matrix contains only **1's** and **0's**. Also, if two or more rows contain the **minimum number of 1's**, the answer is the **lowest of those indices**.

```
class Solution {  
    int minRow(int mat[][]){  
        int ans = -1;  
        int res = 0;  
        for(int i=0; i<mat.length; i++){  
            int count = 0;  
            for(int j=0; j<mat[i].length; j++){  
                if(mat[i][j] == 1) count++;  
            }  
            if(ans == -1){  
                ans = count;  
                res = 0;  
            }else if(ans > count){  
                res = i;  
                ans = count;  
            }  
        }  
        return res+1;  
    }  
};
```

OUTPUT:

The screenshot shows the GeeksforGeeks online judge interface. The URL in the address bar is geeksforgeeks.org/problems/row-with-minimum-number-of-1s5430/1?page=2&category=Matrix&sortBy=submissions. The page title is "geeksforgeeks.org/problems/row-with-minimum-number-of-1s5430/1?page=2&category=Matrix&sortBy=submissions". The top navigation bar includes links for Courses, Tutorials, Practice, and Jobs. The main area shows the "Output Window" tab selected. The "Compilation Results" section indicates "Compilation Completed". A "Case 1" section shows the input values: n=4, m=4, and mat=[[1,1,1,0], [1,1,0,0]]. The output field shows the result 2, and the expected output field also shows 2. The code editor on the right contains the provided Java solution.

```
1- class Solution {  
2-     int minRow(int mat[][]){  
3-         int ans = -1;  
4-         int res = 0;  
5-         for(int i=0; i<mat.length; i++){  
6-             int count = 0;  
7-             for(int j=0; j<mat[i].length; j++){  
8-                 if(mat[i][j] == 1) count++;  
9-             }  
10-            if(ans == -1){  
11-                ans = count;  
12-                res = 0;  
13-            }else if(ans > count){  
14-                res = i;  
15-                ans = count;  
16-            }  
17-        }  
18-        return res+1;  
19-    }  
20-};
```

50. Given a square matrix of size $n \times n$, print the **sum of upper and lower triangular elements**. Upper Triangle consists of elements on the diagonal and above it. The lower triangle consists of elements on the diagonal and below it.

```
class Solution {  
    public static ArrayList<Integer> sumTriangles(int[][] mat) {  
        int n = mat.length;  
        int upperSum = 0;  
        int lowerSum = 0;  
        for (int i = 0; i < n; i++) {  
            for (int j = 0; j < n; j++) {  
                if (j >= i)  
                    upperSum += mat[i][j];  
                if (j <= i)  
                    lowerSum += mat[i][j];  
            }  
        }  
        ArrayList<Integer> ans = new ArrayList<>();  
        ans.add(upperSum);  
        ans.add(lowerSum);  
        return ans;  
    }  
}
```

OUTPUT:

The screenshot shows a browser window displaying a GeeksforGeeks problem page. The URL in the address bar is [geeksforgeeks.org/problems/sum-of-upper-and-lower-triangles-1587115621/1?page=2&category=Matrix&sortBy=submissions](https://www.geeksforgeeks.org/problems/sum-of-upper-and-lower-triangles-1587115621/1?page=2&category=Matrix&sortBy=submissions).

The page title is "Sum of upper and lower triangles". The difficulty is marked as "Easy" and accuracy is 71.0%. Submissions are 75K+ and points are 2. Average time is 15m.

The problem statement asks for the sum of upper and lower triangular elements of a square matrix of size $n \times n$. It specifies that the upper triangle includes the diagonal and elements above it, while the lower triangle includes the diagonal and elements below it.

Examples:

Input: $n = 3, \text{mat}[[[6, 5, 4], [1, 2, 5], [7, 9, 7]]]$

Output: [29, 32]

Explanation: The given matrix is

```
6 5 4  
1 2 5  
7 9 7
```

Upper triangular matrix:

```
6 5 4  
2 5  
7
```

Sum of these elements is $6 + 5 + 4 + 2 + 5 + 7 = 29$.

The code editor on the right contains the Java solution provided in the question. The code is as follows:

```
1- class Solution {  
2-     public static ArrayList<Integer> sumTriangles(int[][] mat) {  
3-         int n = mat.length;  
4-         int upperSum = 0;  
5-         int lowerSum = 0;  
6-         for (int i = 0; i < n; i++) {  
7-             for (int j = 0; j < n; j++) {  
8-                 if (j >= i)  
9-                     upperSum += mat[i][j];  
10-                if (j <= i)  
11-                    lowerSum += mat[i][j];  
12-            }  
13-        }  
14-        ArrayList<Integer> ans = new ArrayList<>();  
15-        ans.add(upperSum);  
16-        ans.add(lowerSum);  
17-        return ans;  
18-    }  
19-}  
20-
```

Below the code editor are buttons for "Custom Input", "Compile & Run", and "Submit".