

## PROBLEM SHEET 1 solution 1:

Search...

Three 90 Ending Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Suggest Feedback

Test Cases Passed  
**1115 / 1115**

Attempts: Correct / Total  
**1 / 1**

Accuracy: 100%

Points Scored  
**2 / 2**

Your Total Score: 2 ↑

Time Taken  
**0.84**

Solve Next

```
1 class Solution {
2     public void reverseArray(int[] arr) {
3         for (int i = 0; i < arr.length / 2; i++) {
4             int temp = arr[i];
5             arr[i] = arr[arr.length - 1 - i];
6             arr[arr.length - 1 - i] = temp;
7         }
8     }
9 }
10
```

Java (21) Start Timer

Custom Input Compile & Run Submit

## Solution 2:

Search...

Three 90 Ending Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Suggest Feedback

Test Cases Passed  
**1111 / 1111**

Attempts: Correct / Total  
**1 / 1**

Accuracy: 100%

Points Scored  
**1 / 1**

Your Total Score: 7 ↑

Time Taken  
**0.29**

Solve Next

```
1 import java.util.ArrayList;
2
3 class Solution {
4     public ArrayList<Integer> getMinMax(int[] arr) {
5
6         int min = arr[0];
7         int max = arr[0];
8
9         // Traverse array
10        for (int i = 1; i < arr.length; i++) {
11            if (arr[i] < min) {
12                min = arr[i];
13            }
14            if (arr[i] > max) {
15                max = arr[i];
16            }
17        }
18
19        // Store result in ArrayList
20        ArrayList<Integer> result = new ArrayList<>();
21        result.add(min);
22        result.add(max);
23
24        return result;
25    }
26 }
27
28
```

Java (21) Start Timer

Custom Input Compile & Run Submit

## Solution 3:

Search...

Three 90 Ending Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Suggest Feedback

Test Cases Passed  
**1121 / 1121**

Attempts: Correct / Total  
**1 / 2**

Accuracy: 50%

Points Scored  
**4 / 4**

Your Total Score: 6 ↑

Time Taken  
**0.57**

Solve Next

```
1 class Solution {
2     public int kthSmallest(int[] arr, int k) {
3         int l = 0, r = arr.length - 1;
4
5         while (l <= r) {
6             int p = partition(arr, l, r);
7
8             if (p == k - 1) return arr[p];
9             else if (p > k - 1) r = p - 1;
10            else l = p + 1;
11        }
12        return -1;
13    }
14
15    private int partition(int[] a, int l, int r) {
16        int pivot = a[r], i = l;
17        for (int j = l; j < r; j++) {
18            if (a[j] <= pivot) {
19                int t = a[i]; a[i] = a[j]; a[j] = t;
20                i++;
21            }
22        }
23        int t = a[i]; a[i] = a[r]; a[r] = t;
24        return i;
25    }
26 }
27
28
--
```

Java (21) Start Timer

Custom Input Compile & Run Submit

## Solution 4:

GeeksforGeeks IDE interface showing Solution 4. The problem is solved successfully. The output window displays the following statistics:

- Test Cases Passed: 1111 / 1111
- Attempts: Correct / Total: 1 / 1
- Accuracy: 100%
- Points Scored: 2 / 2
- Time Taken: 1.01

The code is in Java (21) and implements a solution using a HashSet to find the union of two arrays. The code is as follows:

```
1 import java.util.*;
2
3 class Solution {
4     public ArrayList<Integer> findUnion(int a[], int b[]) {
5
6         // HashSet unique elements store karta hai
7         HashSet<Integer> set = new HashSet<>();
8
9         // Add elements of first array
10        for (int i = 0; i < a.length; i++) {
11            set.add(a[i]);
12        }
13
14        // Add elements of second array
15        for (int i = 0; i < b.length; i++) {
16            set.add(b[i]);
17        }
18
19        // Convert set to ArrayList
20        return new ArrayList<>(set);
21    }
22 }
23
```

## Solution 5:

GeeksforGeeks IDE interface showing Solution 5. The problem is solved successfully. The output window displays the following statistics:

- Test Cases Passed: 1115 / 1115
- Attempts: Correct / Total: 1 / 1
- Accuracy: 100%
- Points Scored: 1 / 1
- Time Taken: 0.79

The code is in Java (21) and implements a solution to find the largest element in an array. The code is as follows:

```
1 class Solution {
2     public int largest(int[] arr) {
3
4         int max = arr[0];
5
6         for (int i = 1; i < arr.length; i++) {
7             if (arr[i] > max) {
8                 max = arr[i];
9             }
10        }
11
12        return max;
13    }
14 }
15
```

## Solution 6:

GeeksforGeeks IDE interface showing Solution 6. The problem is solved successfully. The output window displays the following statistics:

- Test Cases Passed: 1115 / 1115
- Attempts: Correct / Total: 1 / 1
- Accuracy: 100%
- Points Scored: 1 / 1
- Time Taken: 1.09

The code is in Java (21) and implements a solution to rotate an array by one position. The code is as follows:

```
1 class Solution {
2     public void rotate(int arr[]) {
3         int n = arr.length;
4
5         int last = arr[n - 1];
6
7         for (int i = n - 1; i > 0; i--) {
8             arr[i] = arr[i - 1];
9         }
10
11        arr[0] = last;
12    }
13 }
14
15
```

## Solution 7:

Search...

Get 80% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Suggest Feedback

Test Cases Passed  
**1120 / 1120**

Attempts : Correct / Total  
**1 / 1**

Accuracy : 100%

Points Scored  
**4 / 4**

Time Taken  
**0.55**

Your Total Score: 15 ↑

Solve Next

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

Custom Input Compile & Run Submit

## Solution 8:

Array < > 🔍

Submit

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 66 / 66 testcases passed

Ayush\_Ishu7 submitted at Feb 05, 2026 20:26

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

45.04 MB | Beats 9.28%

Code

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

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =  
[1,3,5,6]

target =

## Solution 9:

Problem List < > 🔍

Submit

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 63 / 63 testcases passed

Ayush\_Ishu7 submitted at Feb 05, 2026 20:35

Editorial Solution

Runtime

2 ms | Beats 99.15%

Analyze Complexity

Memory

46.93 MB | Beats 76.86%

Code

```
1 import java.util.HashMap;
2
3 class Solution {
4     public int[] twoSum(int[] nums, int target) {
5         HashMap<Integer, Integer> map = new HashMap<>();
6
7         for (int i = 0; i < nums.length; i++) {
8             int complement = target - nums[i];
9
10            if (map.containsKey(complement)) {
11                return new int[] { map.get(complement), i };
12            }
13
14            map.put(nums[i], i);
15        }
16
17        return new int[] {};
18    }
19 }
20
```

Testcase Test Result

## Solution 10:

ae

Search...

Get 90% Refund

Courses

Tutorials

Practice

Jobs

zA

A

Problem

Editorial

Submissions

Comments

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

1120 / 1120

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 19

Solve Next

Time Taken

0.61

Java (21)

Start Timer

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

--

```
class Solution {
    static int minJumps(int[] arr) {
        int n = arr.length;

        if (n == 1) return 0;
        if (arr[0] == 0) return -1;

        int maxReach = arr[0];
        int steps = arr[0];
        int jumps = 1;

        for (int i = 1; i < n; i++) {
            if (i == n - 1) return jumps;

            maxReach = Math.max(maxReach, i + arr[i]);
            steps--;

            if (steps == 0) {
                jumps++;
                if (i >= maxReach) return -1;

                steps = maxReach - i;
            }
        }
    }
}
```

Custom Input

Compile & Run

Submit