1.

Given an integer array arr[] and an integer k, your task is to find and return the kth smallest element in the given array.

Note: The kth smallest element is determined based on the sorted order of the array.

Examples:

Input: arr[] = [10, 5, 4, 3, 48, 6, 2, 33, 53, 10], k = 4

Output: 5

Explanation: 4th smallest element in the given array is 5.

Input: arr[] = [7, 10, 4, 3, 20, 15], k = 3

Output: 7

Explanation: 3rd smallest element in the given array is 7.

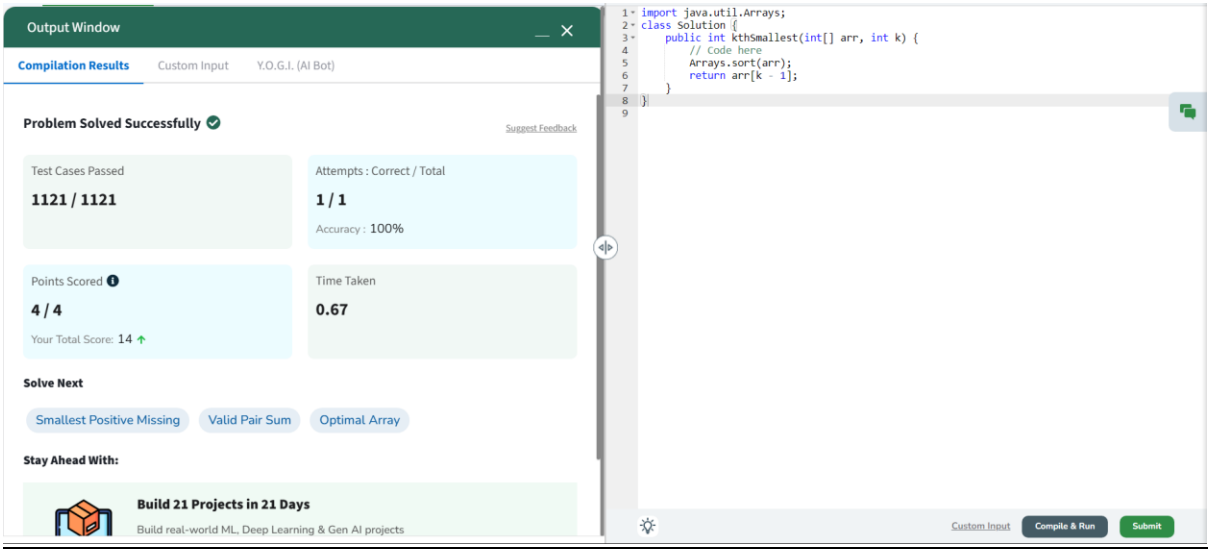
Constraints:

1 ≤ arr.size() ≤ 105

1 ≤ arr[i] ≤ 105

1 ≤ k ≤ arr.size()

Sol



2.

Given an array arr[] denoting heights of n towers and a positive integer k.

For each tower, you must perform exactly one of the following operations exactly once.

Increase the height of the tower by k

Decrease the height of the tower by k

Find out the minimum possible difference between the height of the shortest and tallest towers after you have modified each tower.

You can find a slight modification of the problem here.

Note: It is compulsory to increase or decrease the height by k for each tower. After the operation, the resultant array should not contain any negative integers.

Examples :

Input: k = 2, arr[] = [1, 5, 8, 10]

Output: 5

Explanation: The array can be modified as [1+k, 5-k, 8-k, 10-k] = [3, 3, 6, 8]. The difference between the largest and the smallest is 8-3 = 5.

Input: k = 3, arr[] = [3, 9, 12, 16, 20]

Output: 11

Explanation: The array can be modified as [3+k, 9+k, 12-k, 16-k, 20-k] = [6, 12, 9, 13, 17]. The difference between the largest and the smallest is 17-6 = 11.

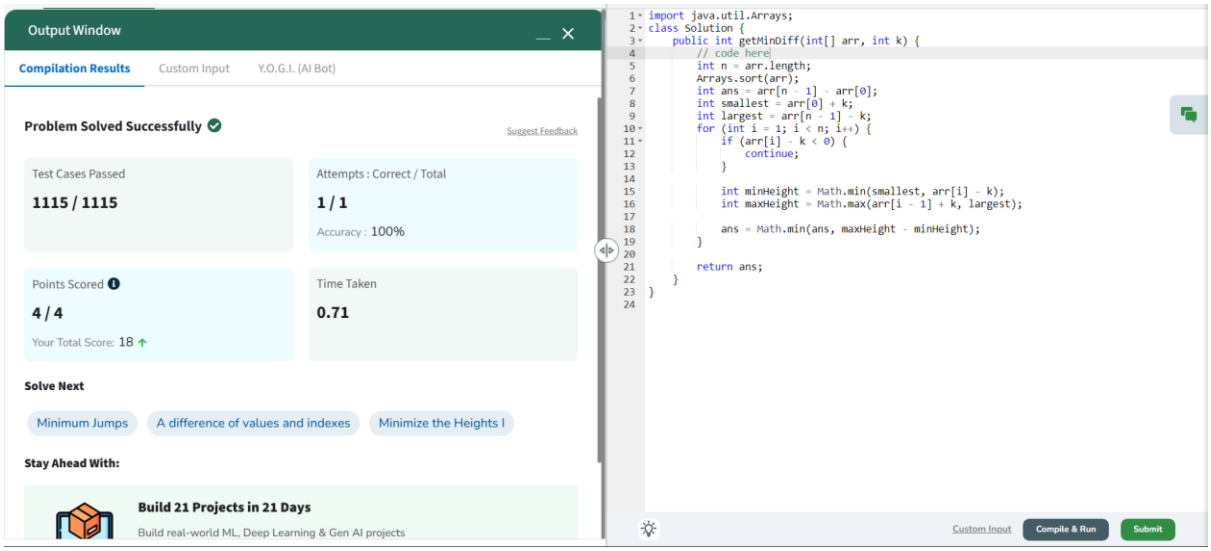
Constraints

$1 \leq k \leq 107$

$1 \leq n \leq 105$

$1 \leq arr[i] \leq 107$

Sol



3.

Given an array arr[] denoting heights of n towers and a positive integer k.

For each tower, you must perform exactly one of the following operations exactly once.

Increase the height of the tower by k

Decrease the height of the tower by k

Find out the minimum possible difference between the height of the shortest and tallest towers after you have modified each tower.

You can find a slight modification of the problem here.

Note: It is compulsory to increase or decrease the height by k for each tower. After the operation, the resultant array should not contain any negative integers.

Examples :

Input: k = 2, arr[] = [1, 5, 8, 10]

Output: 5

Explanation: The array can be modified as [1+k, 5-k, 8-k, 10-k] = [3, 3, 6, 8]. The difference between the largest and the smallest is 8-3 = 5.

Input: k = 3, arr[] = [3, 9, 12, 16, 20]

Output: 11

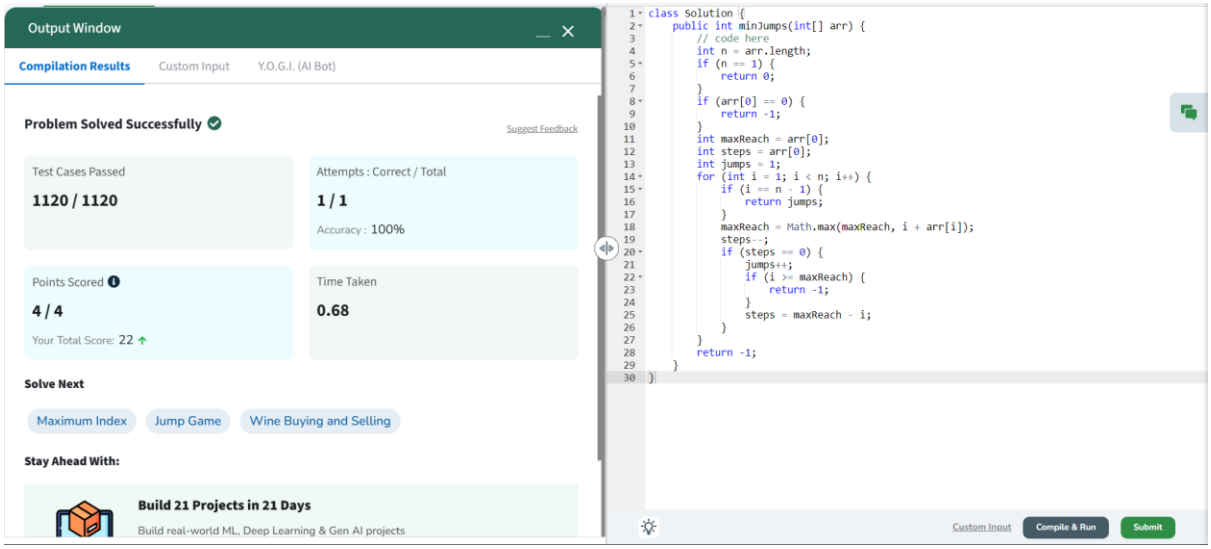
Explanation: The array can be modified as [3+k, 9+k, 12-k, 16-k, 20-k] = [6, 12, 9, 13, 17]. The difference between the largest and the smallest is 17-6 = 11.

Constraints

1 ≤ k ≤ 107

1 ≤ n ≤ 105

1 ≤ arr[i] ≤ 10



4.

Given an integer n, find its factorial. Return a list of integers denoting the digits

that make up the factorial of n.

Examples:

Input: n = 5

Output: [1, 2, 0]

Explanation: 5! = 1*2*3*4*5 = 120

Input: n = 10

Output: [3, 6, 2, 8, 8, 0, 0]

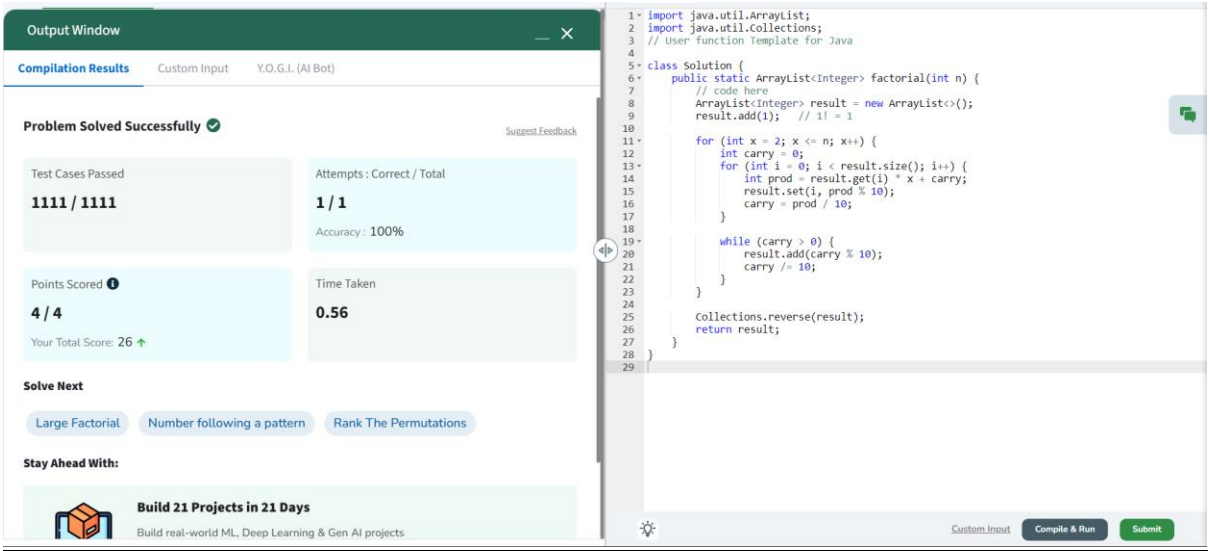
Explanation: 10! = 1*2*3*4*5*6*7*8*9*10 = 3628800

Input: n = 1

Output: [1]

Explanation: 1! = 1

Sol



5.

Given two arrays a[] and b[], your task is to determine whether b[] is a subset

of a[].

Examples:

Input: a[] = [11, 7, 1, 13, 21, 3, 7, 3], b[] = [11, 3, 7, 1, 7]

Output: true

Explanation: b[] is a subset of a[]

Input: a[] = [1, 2, 3, 4, 4, 5, 6], b[] = [1, 2, 4]

Output: true

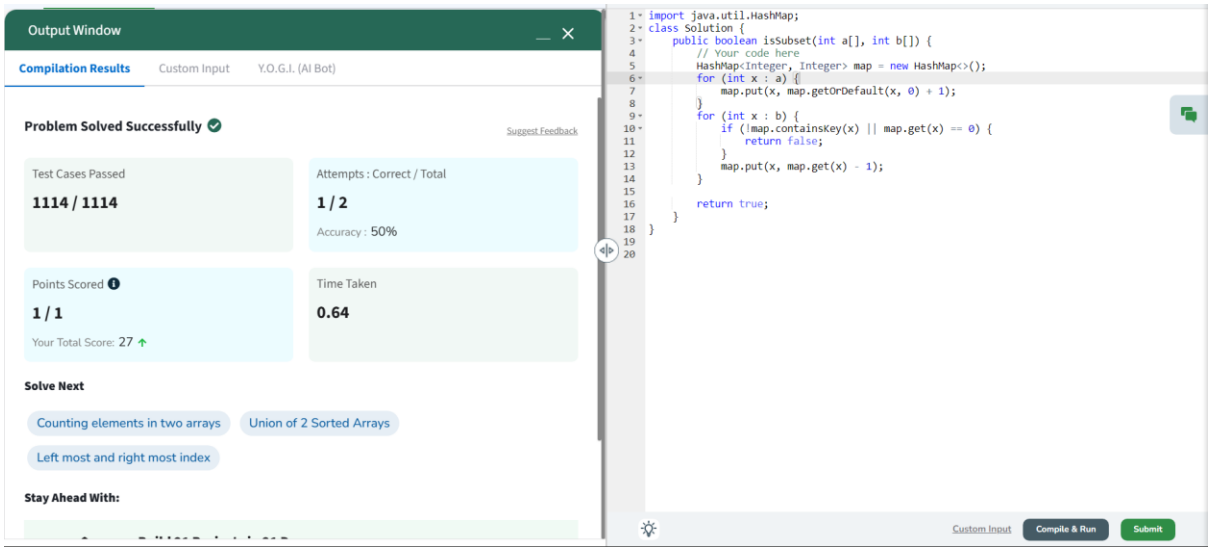
Explanation: b[] is a subset of a[]

Input: a[] = [10, 5, 2, 23, 19], b[] = [19, 5, 3]

Output: false

Explanation: b[] is not a subset of a[]

Sol


Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1114 / 1114
Attempts : Correct / Total
1 / 2
Accuracy : 50%
Points Scored
1 / 1
Your Total Score: 27
Time Taken
0.64
Solve Next
Counting elements in two arrays
Union of 2 Sorted Arrays
Left most and right most index
Stay Ahead With:
import java.util.HashMap;
class Solution {
public boolean isSubset(int a[], int b[]) {
// Your code here
HashMap<Integer, Integer> map = new HashMap<>();
for (int x : a) {
map.put(x, map.getOrDefault(x, 0) + 1);
}
for (int x : b) {
if (!map.containsKey(x) || map.get(x) == 0) {
return false;
}
map.put(x, map.get(x) - 1);
}
return true;
}
}
Custom Input
Compile & Run
Submit