

EY – Coding Assessment

Question 1: Find the Largest and Smallest Number in an Array

You are given an integer array **nums** of size **n**. Your task is to find the **largest** and **smallest** numbers in the array. Return both values as a pair **[smallest, largest]**.

Example 1:

Input: `nums = [4, 7, 2, 9, 1, 5]`

Output: `[1, 9]`

Explanation: The smallest number is 1, and the largest number is 9.

Example 2:

Input: `nums = [10, 10, 10]`

Output: `[10, 10]`

Explanation: All elements are the same, so smallest and largest are both 10.

Constraints:

$1 \leq n \leq 10^4$

$-10^4 \leq \text{nums}[i] \leq 10^4$

Hint: Try solving it in a single pass using $O(1)$ space.

Question 2: Check if a Number is Prime

You are given a number **n**. Your task is to determine whether it is a **prime number** or not. A prime number is one that is greater than 1 and divisible only by 1 and itself.

Example 1:

Input: `n = 7`

Output: `true`

Explanation: 7 is divisible only by 1 and 7.

Example 2:

Input: `n = 12`

Output: `false`

Explanation: 12 is divisible by 1, 2, 3, 4, 6, and 12 — not prime.

Constraints:

$1 \leq n \leq 10^4$

Hint: You don't need to check all numbers up to n — checking up to \sqrt{n} is enough.

Question 3: Count Subarrays Whose Sum is Divisible by K

You are given an array **nums** of integers, and two integers **start** and **end** denoting the range of indices, and a number **k**. Your task is to find the **number of subarrays** within `nums[start..end]` whose sum is **divisible by k**.

Example 1:

Input: nums = [4, 5, 0, -2, -3, 1], start = 0, end = 5, k = 5

Output: 7

Explanation: There are 7 subarrays whose sums are divisible by 5.

Example 2:

Input: nums = [1, 2, 3], start = 0, end = 2, k = 3

Output: 3

Explanation: The subarrays [3], [1, 2], and [1, 2, 3] have sums divisible by 3.

Constraints:

$1 \leq \text{nums.length} \leq 10^4$

$\text{start} \leq \text{end} < \text{nums.length}$

$1 \leq k \leq 10^4$

$-10^4 \leq \text{nums}[i] \leq 10^4$

Hint: Use the prefix sum and remainder approach to solve efficiently in $O(n)$.