Coding Problem: Prime Numbers with Multiple Occurrences

Difficulty Level: Medium Problem considering Hard with Constraints ListArray CodeStub.
Company tag: KPIT

Scenario

In data processing, we often need to extract numbers that occur frequently and have special mathematical properties. Suppose you are given an array of integers, and you are required to find all **prime numbers** that appear more than once. These primes should be collected into a new array for further processing.

Problem Statement

You are given an integer array `A`. Your task is to identify all prime numbers that occur **more than once** in the array and store them in a new array `B`.

- If no prime number repeats, return an empty array.
- The order of elements in `B` should follow their **first appearance** in array `A`.

Input Format

- First line: An integer `N` (size of array `A`).
- Second line: `N` space-separated integers representing the array `A`.

Output Format

- Print the array `B` containing prime numbers that occur more than once.
- If no such prime numbers exist, print `-1`.

Constraints

- 1≤N≤1051 \leq N \leq 10^5
- −106≤A[i]≤106-10^6 \leq A[i] \leq 10^6
- Prime numbers are considered only if they are **greater than 1**.

Sample Input 1

```
10
2 3 5 7 11 3 2 15 17 5
```

Sample Output 1

```
2 3 5
```

Sample Input 2

```
6
4 6 8 9 10 12
```

Sample Output 2

```
-1
```

Explanation

- In Example 1: The array contains primes **2, 3, 5** that occur more than once. Hence, `B = [2, 3, 5]`.
- In Example 2: No prime numbers repeat. Hence, output is `-1`.

Java Code Stub

```
import java.util.*;

public class PrimeDuplicates {

    // Method to check if a number is prime
    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;
    }

    // Method to find prime duplicates
    static List<Integer> findPrimeDuplicates(int[] arr) {
        Map<Integer, Integer> freq = new LinkedHashMap<>();
        for (int num : arr) {
            if (isPrime(num)) {
                freq.put(num, freq.getOrDefault(num, 0) + 1);
            }
        }

        List<Integer> result = new ArrayList<>();
```

```
        for (Map.Entry<Integer, Integer> entry : freq.entrySet()) {
            if (entry.getValue() > 1) {
                result.add(entry.getKey());
            }
        }
        return result;
    }

    // Main Method
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] A = new int[N];
        for (int i = 0; i < N; i++) {
            A[i] = sc.nextInt();
        }

        List<Integer> result = findPrimeDuplicates(A);
        if (result.isEmpty()) {
            System.out.println("-1");
        } else {
            for (int num : result) {
                System.out.print(num + " ");
            }
        }
    }
}
```