Lab 2

Problem 1

You are given two sorted arrays A and B of sizes m and n, respectively. Your task is to return the **median** of the two sorted arrays. The overall run time complexity of your solution must be $\mathcal{O}(\log(m + n))$.

Definition of Median

- If the total number of elements $(m + n)$ is odd, the median is the middle element in the combined sorted array.
- If $(m+n)$ is even, the median is the average of the two middle elements in the combined sorted array.

Input

- The first line contains two integers m and n — the sizes of arrays A and B.
- The second line contains m integers, the elements of array A, in non-decreasing order.
- The third line contains n integers, the elements of array B, in non-decreasing order.

Output

Output a single number — the median of the two sorted arrays.

Example 1

Input

```
2 1
1 3
2
```

Output

```
2
```

Example 2

Input

```
2 2
1 2
3 4
```

Output

```
2.5
```

Constraints

- Arrays A and B are sorted in non-decreasing order.
- Time complexity must be $\mathcal{O}(\log(m+n))$.

Note

A direct merge of both arrays would take $\mathcal{O}(m+n)$ time, which is not acceptable. Instead, use a binary search approach to partition the arrays such that the left half and right half satisfy the median conditions.

Problem 2

Let A be an $n \times n$ matrix of integers such that each row and each column are arranged in **ascending order**. That is:

$$A[i][0] \leq A[i][1] \leq \cdots \leq A[i][n-1] \quad \text{for all rows } i$$

$$A[0][j] \leq A[1][j] \leq \cdots \leq A[n-1][j] \quad \text{for all columns } j$$

Given an integer k, determine whether k appears in A. If k is present, output its position (i, j) such that $A[i][j] = k$ (0-based indexing). Otherwise, output `-1 -1`.

Input Format

- The first line contains an integer n — the dimension of the matrix.
- The next n lines each contain n integers, representing the matrix A.
- The last line contains an integer k — the target value to search for.

Output Format

- If k is present, output two integers i and j (0-based) — the row and column index.
- If k is not present, output `-1 -1`.

Example

Input

```
4
1   4   7  11
2   5   8  12
3   6   9  16
10  13  14  17
5
```

Output

```
1 1
```

Explanation

The target 5 is located at row 1, column 1 (0-based indexing).

Desired Time Complexity

$$\mathcal{O}(n) \quad \text{time.}$$

Problem 3

You are given a mountain array — an array of n distinct integers that is strictly increasing up to a single peak, then strictly decreasing. Your task is to find the index of a target value x in the mountain array. If x does not exist, return -1.

Input Format

- The first line contains an integer n, the size of the mountain array.
- The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$, representing the mountain array.
- The third line contains the target integer x.

Output Format

Output the index of x if it exists, otherwise output -1.

Example

Input

```
7
1 3 5 7 6 4 2
4
```

Output

```
5
```

Explanation: The array is strictly increasing until index 3 (where 7 is the peak), then strictly decreasing. The target 4 is found at index 5.

Constraints

- The array is a valid mountain array: there exists an index p $(0 < p < n-1)$ such that

$$a_0 < a_1 < \cdots < a_p \quad \text{and} \quad a_p > a_{p+1} > \cdots > a_{n-1}.$$

- Time complexity: $\mathcal{O}(\log n)$.