**Problem 1: Library Bot's Recursive Quest**

Deep inside a sprawling digital archive of rare manuscripts, a tireless AI named LibBot assists researchers worldwide. LibBot maintains an ordered list of digital catalog IDs representing unique documents. One day, a historian urgently needs to find a particular text buried in this massive archive. The list is already sorted, but due to data privacy protocols, LibBot must use a recursive lookup technique to avoid scanning all entries linearly, preserving computational efficiency.

Your job is to implement this recursive search method that mimics LibBot’s logic, ensuring the historian gets the correct document as fast as possible.

**Input Format:**

- First line: Integer `n` (number of books)

- Second line: `n` space-separated sorted book IDs

- Third line: Integer `x` (book ID to search)

**Output Format:**

- Index of the book if found, else -1

**Sample Input:**

5

10 20 30 40 50

30

**Sample Output:**

2

**Sample Input (Not Found):**

5

10 20 30 40 50

35

**Sample Output:**

-1

**Problem 2: The Sorting Robot – Dutch Flag System**

In a fully automated warehouse powered by AI, robots sort colored marbles that represent different priority packages: red (0) for urgent, white (1) for normal, and blue (2) for low priority. These marbles move chaotically along conveyor belts, and the robot must organize them quickly and accurately using minimum memory and swaps.

The company challenges you to simulate the robot’s sorting algorithm known as the Dutch National Flag problem to streamline their shipping operations.

**Input Format:**

- First line: Integer `n`

- Second line: `n` space-separated integers (only 0, 1, 2)

**Output Format:**

- Sorted array

**Sample Input:**

6

2 0 2 1 1 0

**Sample Output:**

0 0 1 1 2 2

**Problem 3: AI's Disorder Meter**

A tech startup is building an AI that evaluates online product reviews by how closely the review order matches their product rating scores. The less ordered the list is, the more inversions it contains. An inversion is a scenario where a higher score precedes a lower one. This count helps determine whether the platform is displaying products fairly.

You have been hired to design the component that calculates this inversion count to assess the system's disorder level.

**Input Format:**

- First line: Integer `n`

- Second line: `n` space-separated integers

**Output Format:**

- Number of inversions

**Sample Input:**

5

2 4 1 3 5

**Sample Output:**

3

**Sample Input (Fully Sorted):**

5

1 2 3 4 5

**Sample Output:**

0

**Problem 4: Syncing Event Timelines**

In a popular collaborative scheduling app, two teammates have been planning their daily tasks separately. At the end of the day, the backend system must merge these two chronologically sorted timelines into one cohesive view. Due to tight memory limits on low-end devices, the merging must be done in-place without allocating extra space.

You are assigned to build this in-place merge feature to ensure both timelines are reflected accurately on the shared dashboard.

**Input Format:**

- First line: Integer `n` (size of first array)

- Second line: `n` space-separated integers (sorted)

- Third line: Integer `m` (size of second array)

- Fourth line: `m` space-separated integers (sorted)

**Output Format:**

- Merged sorted list

**Sample Input:**

4

1 3 5 7

3

2 4 6

**Sample Output:**

1 2 3 4 5 6 7

**Problem 5: Signal on the Mountain Top**

A defense-grade surveillance drone flies over a rugged mountain range capturing elevation data. Engineers use this data to identify peaks—specific points that are higher than their immediate neighbors. Peaks often correspond to strategic locations for installing signal towers or weather stations.

You’ve been asked to help process this elevation data by quickly identifying at least one peak. But the drone is battery-limited and cannot afford linear scans over large regions, so your method must be efficient.

**Input Format:**

- First line: Integer `n`

- Second line: `n` space-separated integers

**Output Format:**

- One peak element

**Sample Input:**

6

1 3 20 4 1 0

**Sample Output:**

20