**Problem 1: The Case of the Missing Key**

Ravi has just returned from college and is rushing to get ready for his evening football game. He tosses his bag on the bed and begins searching for the key to his bicycle lock. The room is a mess—books on the table, clothes on the chair, gadgets on the floor. He recalls that he might have dropped the key somewhere around the corners while unpacking.

His plan is simple: he decides to check every spot he can think of, one by one, in the same order he usually drops things.

Given a list of possible places Ravi checks and the item he is looking for, help determine where exactly the key is found (if at all).

**Input Format:**

* First line: An integer n representing the number of places he checks.
* Second line: n space-separated integers representing the unique code assigned to each place.
* Third line: An integer x representing the code of the location where the key might be.

**Output Format:**

* Index (0-based) of the location if the key is found; otherwise, -1.

**Sample Input:**

5

101 203 305 407 509

305

**Sample Output:**

2

**Sample Input:**

5

101 203 305 407 509

999

**Sample Output:**

**-1**

**Problem 2: The Librarian's Puzzle**

A historic library is known for its centuries-old system of cataloging books using numeric tags. Recently, a digital transformation is underway. To test the new system, the head librarian selects a sorted list of book tag numbers from a shelf and wants to quickly locate a particular volume that was misplaced.

She feeds the sorted list into the system and enters the ID of the book she wants to find. The system must return the exact position of the book on the shelf or confirm it is missing.

Given the catalog and the target book's ID, help the librarian identify the correct position.

**Input Format:**

* First line: Integer n representing number of books
* Second line: n space-separated sorted integers representing the catalog numbers
* Third line: Integer key (book ID to find)

**Output Format:**

* Index of the book if found; otherwise, -1

**Sample Input:**

6

11 22 33 44 55 66

44

**Sample Output:**

3

**Sample Input:**

6

11 22 33 44 55 66

77

**Sample Output:**

-1

**Problem 3: Counting the Voter**

During the state elections, a team is tasked with validating the turnout of citizens using their unique voter ID numbers. The IDs were collected digitally from multiple booths across the city and merged into one dataset.

A specific citizen claims to have voted multiple times due to a technical error. To verify this claim, the team needs to count how many times this citizen's voter ID appears in the list.

Can you assist the election team in verifying this claim by counting the occurrences of a given voter ID?

**Input Format:**

* First line: Integer n (number of entries)
* Second line: n space-separated integers (voter IDs)
* Third line: Integer x (voter ID to count)

**Output Format:**

* Integer representing the number of occurrences

**Sample Input:**

7

101 105 101 103 101 102 104

101

**Sample Output:**

3

**Sample Input:**

7

101 105 101 103 101 102 104

999

**Sample Output:**

0

**Problem 4: Inventory Inspection**

An online grocery warehouse updates its stock prices daily. The warehouse manager suspects the update script might be introducing inconsistencies. To ensure everything is running smoothly, you are given a snapshot of the current prices for a specific category of products.

You need to examine whether the price list was updated in the correct order. If any price is lower than the one before it, the update process must have malfunctioned.

Help the manager by verifying the sequence of prices.

**Input Format:**

* First line: Integer n (number of items)
* Second line: n space-separated integers representing item prices

**Output Format:**

* Print Yes if sorted in ascending order; otherwise, print No

**Sample Input:**

5

10 20 30 40 50

**Sample Output:**

Yes

**Sample Input:**

5

10 30 20 40 50

**Sample Output:**

No

**Problem 5: Neha's Score Card**

Neha is preparing her personal progress report using scores she received from mock tests over the past month. She records them all at once, but now she wants to display the scores in a neat, ascending order before pasting them into her diary.

Since she prefers to sort them by a method she recently learned from her tutor, she asks for your help to reorder the scores manually.

Can you help Neha sort her scores for the report?

**Input Format:**

* First line: Integer n (number of scores)
* Second line: n space-separated integers

**Output Format:**

* Sorted scores in ascending order (space-separated)

**Sample Input:**

4

5 1 4 2

**Sample Output:**

1 2 4 5

**Sample Input:**

4

1 2 3 4

**Sample Output:**

1 2 3 4

**Problem 6: Ranking the Batch**

The school administration has gathered final exam marks for a batch of students. Unfortunately, due to a system migration, the marks are currently stored in an unsorted list. Before announcing the toppers and organizing the convocation ceremony, the administration wants to publish an official rank list from the lowest to the highest score.

To maintain transparency and fairness, every student's name will appear on the rank list in sorted order of their marks. You’ve been given the responsibility of helping the administration arrange the marks correctly.

**Input Format:**

* First line: Integer n (number of students)
* Second line: n space-separated integers (marks)

**Output Format:**

* Sorted marks in ascending order (space-separated)

**Sample Input:**

5

50 20 40 10 30

**Sample Output:**

10 20 30 40 50

**Sample Input:**

4

50 50 50 50

**Sample Output:**

50 50 50 50

**Problem 7: Organizing Grades in Real-Time**

Ms. Nisha is teaching a live online test-prep class. Students submit their scores one by one as they complete the quiz. The grading software she uses updates the result sheet dynamically by inserting each new score into its correct position, ensuring the sheet is always sorted.

At the end of the quiz, she wants to display the final sorted score list. Your task is to simulate this real-time sorting process and provide the list as it would appear after all entries have been processed.

**Input Format:**

* First line: Integer n (number of scores)
* Second line: n space-separated integers (stream of scores)

**Output Format:**

* Sorted scores after all inputs (space-separated)

**Sample Input:**

6

60 10 40 30 90 50

**Sample Output:**

10 30 40 50 60 90

**Problem 8: Leaderboard Update**

The coding competition platform "CodeChamp" has a dynamic leaderboard that updates every time a participant submits a solution. To keep the leaderboard sorted, each new score must be inserted in the right place.

A new contestant just submitted a score, and before it officially appears on the board, we want to predict where it would rank.

Given the current sorted leaderboard scores, determine the index at which the new score would be inserted.

**Input Format:**

* First line: Integer n (number of existing scores)
* Second line: n space-separated integers (sorted scores)
* Third line: Integer x (new score)

**Output Format:**

* Index at which the score should be inserted

**Sample Input:**

5

10 20 30 40 50

35

**Sample Output:**

3

**Problem 9: Rotating Time Slots**

An employee attendance management system rotates login time slots every week to balance network load. After a few weeks, the system ends up with a rotated version of a sorted schedule.

Although the list still contains all time slots, the order is shifted. The system needs to identify the earliest possible login time from the current list to allocate for a new joiner.

Write a program to find the smallest login time in a rotated sorted list.

**Input Format:**

* First line: Integer n (number of login times)
* Second line: n space-separated integers (rotated sorted times)

**Output Format:**

* Minimum login time

**Sample Input:**

6

40 50 60 10 20 30

**Sample Output:**

10

**Problem 10: Timestamp Traceback**

A server log records timestamps of access requests throughout the day. To audit a specific incident, the cybersecurity team needs to determine the exact timeframe during which a certain timestamp occurred repeatedly.

You’re given a sorted list of timestamps from the log, where a particular timestamp may appear multiple times consecutively. Help the team trace the first and last occurrence of a specific timestamp.

**Input Format:**

* First line: Integer n (number of timestamps)
* Second line: n space-separated sorted integers (timestamps)
* Third line: Integer x (timestamp to find)

**Output Format:**

* Two integers representing the first and last index of x

**Sample Input:**

8

2 4 4 4 6 7 9 9

4

**Sample Output:**

1 3

**Sample Input:**

8

2 4 4 4 6 7 9 9

5

**Sample Output:**

-1 -1