Problem - 01

Problem Statement

You are given a sorted array of **distinct** integers **nums** and a target value **target**. Your task is to return the index if the target is found. If not, return the index where it would be inserted in order.

You must implement an algorithm with **O(log n)** runtime complexity.

Input

- The first line contains an integer **n** — the number of elements in the array.
- The second line contains **n** integers **nums[i]** representing the sorted array in non-decreasing order.
- The third line contains a single integer **target** — the value to find in the array.

Output

- Print a single integer idx, the index where target is found or should be inserted.

Test Cases

Input	Output
4 1 3 5 6 5	2
4 1 3 5 6 2	1
4 1 3 5 6 7	4

Problem - 02

Problem Statement

Given an array of integer nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If the target is not found in the array, print [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

Input

- The first line contains an integer **n** — the number of elements in the array.
- The second line contains **n** integers **nums[i]** representing the sorted array in non-decreasing order.
- The third line contains a single integer **target** — the value to find in the array.

Output

Print two space-separated integers l r, where:

- l is the first occurrence index of the target value in the array.
- r is the last occurrence index of the target value in the array.
- If the target does not exist, print "-1 -1".

Test Cases

Input	**Output**
6 5 7 7 8 8 10 8	3 4
6 5 7 7 8 8 10 6	-1 -1
0 0	-1 -1

Problem - 03

Problem Statement

You are given two strings **s** and **t** consisting of lowercase English letters. Your task is to determine whether **t** is an anagram of **s**.

A string **t** is an anagram of **s** if it is formed by rearranging the letters of **s** without adding or removing any characters.

Input

- The first line contains a string **s** — the first string.
- The second line contains a string **t** — the second string.
- Both **s** and **t** consist only of lowercase English letters (**'a'** to **'z'**).

Output

- Print "YES" if t is an anagram of s, otherwise print "NO".

Test Cases

Input	**Output**
anagram nagaram	YES
rat car	NO

Problem - 04

Problem Statement

You are given an array **nums** containing **n** distinct numbers in the range **[0, n]**. Your task is to find and return the only number in this range that is missing from the array.

Input

- The first line contains an integer **n** — the number of elements in the array.
- The second line contains **n** distinct integers **nums[i]** (**0 <= i < n**) representing the given array.

Output

- Print a single integer — the missing number in the range [0, n].

Test Cases

Input	Output
3 3 0 1	2
4 0 1 2 4	3

Problem - 05

Problem Statement

You are given a sorted array **nums** containing **n** distinct numbers in the range **[0, n]** in strictly increasing order. Your task is to find and return the only number in this range that is missing from the array.

You must implement an algorithm with **O(log n)** runtime complexity.

Input

- The first line contains an integer **n** — the number of elements in the array.
- The second line contains **n** distinct integers **nums[i]** (**0 <= i < n**) representing the given sorted array in strictly increasing order.

Output

- Print a single integer — the missing number in the range [0, n].

Test Cases

Input	**Output**
3 0 1 3	2
4 0 1 2 4	3

Problem - 06

Problem Statement

You are given a positive integer **num**. Your task is to determine whether **num** is a perfect square.

A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself.

You **must not** use any built-in library functions such as **sqrt**. You must implement an algorithm with **O(log n)** runtime complexity.

Input

- The first and only line contains a single integer **num**.

Output

- Print **"YES"** if **num** is a perfect square, otherwise print **"NO"**.

Test Cases

Input	**Output**
16	YES
14	NO

Problem - 07

Problem Statement

You are given a non-negative number **N**, and your task is to calculate its square root to a precision of 10^{-3}.

The square root of a number is a value that, when multiplied by itself, gives the original number.

Input

- The input consists of a single floating-point number **N.**

Output

- Output the square root of **N**, rounded to 3 decimal places.

Test Cases

Input	**Output**
16	4.000
2	1.414
0	0.000