Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 10

Allowed languages: **C, C++**

General Tips

- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDE or text editors like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. However, in the lab you will be using command line interface.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- You can implement your solutions in C or C++. Using C++ STL (Standard Template Library) is allowed.

Problem A. Maximum Subarray Sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Given an integer array a of size n, find the subarray with the largest sum, and return its sum.

Note:

A subarray is a contiguous non-empty sequence of elements within an array.

Assume 1-based indexing.

The time complexity of your solution should be $O(nlogn)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^5$) — the size of array a.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

Output

Print one line containing the maximum subarray sum of a.

Example

standard input	standard output
9 -2 1 -3 4 -1 2 1 -5 4	6

Explanation

The subarray $[4, -1, 2, 1]$ has the largest sum 6. It can be shown that no other subarray has sum greater than 6.

Problem B. Search Matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Write an algorithm that searches for a value v in an $m \times n$ integer matrix a. This matrix has the following properties:

- Integers in each row are sorted in ascending order from left to right.
- Integers in each column are sorted in ascending order from top to bottom.

Note:

Assume 1-based indexing. Note that the matrix can have duplicate entries.

The time complexity of your solution should be $O(m+n)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains two integers, m and n ($1 \leq m, n \leq 10^5$) — the dimensions of matrix a. m and n are not necessarily equal.

The following m lines contain the rows of a. The i-th line contains n space separated integers `a[i][1], a[i][2], ..., a[i][n]` ($-10^9 \leq$ `a[i][j]` $\leq 10^9$) — the elements of the i-th row of matrix a.

The last line of input contains v ($-10^9 \leq v \leq 10^9$) — the value to be searched for.

Output

Print `YES` if the target value, v, is present in the matrix a and `NO` otherwise.

Examples

standard input	standard output
5 5 1 4 7 11 15 2 5 8 12 19 3 6 9 16 22 10 13 14 17 24 18 21 23 26 30 5	YES

Explanation

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Problem C. Median of Two Sorted Arrays

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Let a and b be two sorted integer arrays of size n and m respectively, and find the median of the arrays a and b.

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(log(n + m))$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^5$) — the size of array a.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

The first line of input contains m ($1 \leq m \leq 10^5$) — the size of array b.

The second line of input contains m space separated integers $b_1, b_2, \ldots, b_m$ ($-10^9 \leq b_i \leq 10^9$) — the elements of array b.

Output

Print one line containing the median of arrays a and b.

Examples

standard input	standard output
2 1 3 1 2	2
2 1 2 2 3 4	2.50000

Explanation

In example 1, merged array = $[1, 2, 3]$ and median is 2.

In example 2, merged array = $[1, 2, 3, 4]$ and median is $(2 + 3)/2 = 2.5000$.

Problem D. Pancake Sorting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Given an array, a of n integers, sort the array by performing a series of pancake flips and return an array of the k-values corresponding to a sequence of pancake flips that sort a. Your answer must sort the array within $10n$ flips.

Note:

A subarray is a contiguous non-empty sequence of elements within an array.

Assume 1-based indexing.

In one pancake flip we do the following steps:

- Choose an integer k where $1 \leq k \leq n$.
- Reverse the sub-array `a[1...k]`.

For example, if $a = [3, 2, 1, 4]$ and we performed a pancake flip choosing $k = 3$, we reverse the sub-array $[3, 2, 1]$, so $a = [1, 2, 3, 4]$ after the pancake flip at $k = 3$.

The time complexity of your solution should be $O(n^2)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^5$) — the size of array a.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of array a.

It is guaranteed that the elements of a are unique (i.e., a is a permutation of the integers from 1 to n).

Output

The first line of output should contain m ($0 \leq m \leq 10n$), the number of flips you perform to sort array a.

The second line of output should contain m space-separated integers — the i-th integer should correspond to the k value for the i-th flip.

Examples

standard input	standard output
4 3 2 1 4	1 3
4 3 4 2 1	2 2 4

Explanation

In example 2,

- The array after the first flip with $k = 2$: $[4, 3, 2, 1]$.
- The array after the second flip with $k = 4$: $[1, 2, 3, 4]$.

Problem E. mkv

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array a containing n pairs of integers where $a_i = \{w_i, h_i\}$ represents the width and height of a Russian doll.

The i-th doll can fit inside the j-th doll if and only if both the width and height of the j-th doll are strictly greater than that of the i-th doll.

Find the maximum number of dolls that you can fit inside one another in a nested manner.

Note:

Assume 1-based indexing.
If none of the dolls fit inside another doll then the output will be one.

The time complexity of your solution should be $O(nlogn)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^5$) — the size of array a.

The following n lines of input contain the elements of a. The i-th line contains two space separated integers, w_i and h_i ($1 \leq w_i, h_i \leq 10^5$) — the width and height of the i-th doll.

Output

Print one line containing the maximum number of dolls that you can fit inside one another in a nested manner.

Example

standard input	standard output
4 5 4 6 4 6 7 2 3	3

Explanation

The following 3 dolls can fit inside one another in a nested manner: $[2, 3] \rightarrow [5, 4] \rightarrow [6, 7]$.

Problem F. Disorder

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array A of distinct non-negative integers of size n. Two elements A_i and A_j are said to be out of order if $i < j$ and $A_i > A_j$. For example, consider the array A = [2,3,4,1]. Here, A_0, i.e., 2 and A_3, i.e., 1 are out of order (3,1 and 4,1 are also out of order).

Find the number of pairs that are out of order. In the above example, there are 3 pairs that are out of order.

The time complexity of your solution should be $O(n * log(n))$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of a single integer n ($1 \leq n \leq 10^5$) — the size of the array A.

The second line of input contains n space-separated integers $A_1, A_2, \ldots, A_n$ ($0 \leq A_i \leq 10^9$) — the elements of the array.

Elements of A are distinct.

Output

Print a single integer, the number of pairs that are out of order.

Examples

standard input	standard output
5 2 4 1 3 5	3
5 2 3 4 5 6	0

Note

In the first example the pairs that are out of order are (2,1) (4,1) and (4,3).
Note that above the elements of the array are given, not the indices

In the second example, the array is sorted, so there are no pairs that are out of order.

Problem G. ICL

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The Indian Cricket League (ICL) is a new knockout style cricket tournament organized by the Indian Cricket Board. There's a match tomorrow between the two most successful teams Mumbai Mavericks and Chennai Cheetahs. The head coach of the Cheetahs is analyzing the batting stats of the Mavericks. He has stats of runs scored by all n players in the previous season in the form of a list and he wants to find the maximum score for each and every contiguous sub-list of size k to strategize for the game.

The time complexity of your solution should be $O(n)$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of two integers n and k ($1 \leq k \leq n \leq 10^5$) — the number of players and the size of the sub-list respectively

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) — the runs scored by each player

Output

Print $n - k + 1$ integers - the maximum runs of every contiguous subarray of size k.

Examples

standard input	standard output
9 3 1 2 3 1 4 5 2 3 6	3 3 4 5 5 5 6
10 4 8 5 10 7 9 4 15 12 90 13	10 10 10 15 15 90 90

Note

In the first example,

- 1st contiguous subarray = {1 2 3} Max = 3
- 2nd contiguous subarray = {2 3 1} Max = 3
- 3rd contiguous subarray = {3 1 4} Max = 4
- 4th contiguous subarray = {1 4 5} Max = 5
- 5th contiguous subarray = {4 5 2} Max = 5
- 6th contiguous subarray = {5 2 3} Max = 5
- 7th contiguous subarray = {2 3 6} Max = 6

In the second example,

- 1st contiguous subarray = {8 5 10 7}, Max = 10
- 2nd contiguous subarray = {5 10 7 9}, Max = 10
- 3rd contiguous subarray = {10 7 9 4}, Max = 10

- 4th contiguous subarray = {7 9 4 15}, Max = 15
- 5th contiguous subarray = {9 4 15 12}, Max = 15
- 6th contiguous subarray = {4 15 12 90}, Max = 90
- 7th contiguous subarray = {15 12 90 13}, Max = 90

Problem H. Dependency Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given two arrays A and B of size n and m respectively. Sort the elements of the first array A such that all the relative positions of the elements in A are the same as the elements in B. Note that the elements of B are distinct while A may contain repeated elements.

For the elements of A that are not in B, append them to the end in increasing order.

The time complexity of your solution should be $O(n * log(n) + m * log(m))$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of two integers n and m ($1 \leq n, m \leq 10^5$) — the size of A and B respectively. Note that m can be less than, equal to, or greater than n.

The second line of input contains n space-separated integers $A_1, A_2, \ldots, A_n$ ($1 \leq A_i \leq 10^6$) — the elements of A

The third line of input contains m space-separated integers $B_1, B_2, \ldots, B_m$ ($1 \leq B_i \leq 10^6$) — the elements of B. These elements are distinct.

Output

Print n integers - the elements of A sorted based on the rules given above

Examples

standard input	standard output
11 4 2 1 2 5 7 1 9 3 6 8 8 2 1 8 3	2 2 1 1 8 8 3 5 6 7 9
11 4 2 1 2 5 7 1 9 3 6 8 8 99 22 444 56	1 1 2 2 3 5 6 7 8 8 9

Note

In the first example, 2 comes first then 1 comes, then comes 8, then finally 3 comes. We append the remaining elements in sorted order.

In the second example, no element of A occurs in B, so we sort the elements of A in ascending order.

Problem I. Impartial Swap

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You distribute chocolates to your two children, such that the first child gets n chocolates and the second child gets m chocolates. Each chocolate is associated with some tastiness level. The satisfaction of a child is defined as the sum of the tastiness of the chocolates given to them.

You are given the tastiness of each chocolate given to a child in the form of an array. Check if it is possible for the children to have equal satisfaction if you are allowed to **swap at most one pair of chocolates** between them.

The time complexity of your solution should be $O(n * log(n) + m * log(m))$. Any solution with a higher time complexity will not be considered.

Input

The first line of input consists of two integers, n and m ($1 \leq n, m \leq 10^5$)— the number of chocolates given to the first and the second child respectively.

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the tastiness level of the chocolates given to the first child.

The third line of input contains m space-separated integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 10^9$) — the tastiness level of the chocolates given to the second child

Output

Print Yes if it is possible to get the same satisfaction level and No if it is not possible.

Examples

standard input	standard output
6 4 4 1 2 1 1 2 3 6 3 3	Yes
4 4 5 7 4 6 1 2 3 9	No

Note

In the first example,

- Sum of elements in $A = 11$
- Sum of elements in $B = 15$,
- To get the same sum from both arrays, we can swap the following values: 1 from A and 3 from B. Note that we swapped only one pair from A and B, i.e., we took one from A and gave it to B and took one from B and gave it to A. After the swap, the number of chocolates for both A and B remains same as before.

In the second example, no matter what elements we swap, we can not get the same satisfaction.

Problem J. Matrix Search II

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Write an algorithm to find the k-th smallest element in an $n \times n$ matrix. This matrix has the following properties:

- Integers in each row are sorted in ascending order from left to right.
- Integers in each column are sorted in ascending order from top to bottom.

Note that the matrix can have duplicate entries.

The time complexity of your solution should be $O((n+k) * log(n))$ after ignoring the time to take the input. Any solution with a higher time complexity will not be considered.

Input

The first line consists of two integers n ($1 \leq n \leq 1000$) and k ($1 \leq k \leq n * n$) — where n is the dimension of the matrix

Then n lines follow, the i-th line consists of n space separated integers $a_{i1}, a_{i2}, \ldots, a_{in}$ ($1 \leq a_{ij} \leq 10^9$) — the elements of the i-th row of the matrix.

Output

Print a single integer the k-th smallest element of the matrix. Assume 1-based indexing to determine k-th.

Example

standard input	standard output
4 7 10 20 30 40 15 25 35 45 24 29 37 48 32 33 39 50	30
4 3 16 28 60 64 22 41 63 91 27 50 87 93 36 78 87 94	27