**Problem**

A shop has a stack of chocolate boxes each containing a positive number of chocolates. Initially, the stack is empty. During the next N minutes, either of these two things may happen:

* The box of chocolates on top of the stack gets sold
* You receive a box of chocolates from the warehouse and put it on top of the stack.

Determine the number of chocolates in the sold box each time he sells a box.

**Notes**

* If C[i] = 0, he sells a box. If C[i] > 0, he receives a box containing C[i] chocolates.
* It is confirmed that he gets a buyer only when he has a non-empty stack.
* The capacity of the stack is infinite.

**Example 1**

*Assumptions*

***Input***

* *N = 4*
* *C = [2, 1, 0, 0]*

***Output:***1 2

*Approach*

After the first two minutes, the stack is [1, 2].

During the third minute, the box on the top having 1 chocolate is sold.

During the fourth minute, the box on the top having 2 chocolates is sold.

**Function description**

Complete the function ***solve()***provided in the editor. The function takes the following 2 parameters and returns the solution.

* *N*: Represents the number of minutes
* *C*: Represents the description of boxes

**Input format for custom testing**

**Note:** Use this input format if you are testing against custom input or writing code in a language where we don’t provide boilerplate code

* The first line contains *N*denoting the number of minutes.
* The second line contains *C*denoting the array consisting of the box descriptions.

**Output format**

Print an array, representing the number of chocolates in the sold box each time you sell a box.

**Constraints**

1≤N≤105

0≤c[i]]≤109

**Sample Input**

3

5 0 5

**Sample Output**

5

**Explanation**

*Given*

***Input***

* *N = 3*
* *C = [5, 0, 5]*

***Output:***5

*Approach*

After the first minute, the stack is [5].

During the second minute, the top of the stack has 5 chocolates.

After the third minute, the stack is [5].

James visits a restaurant, looks at the menu, and realizes that there is no price on it. Since he wanted to know the prices before he orders, he looked up the restaurant online and found n different versions of the menu. He knew from experience that usually the menu which has the maximum number of items that have the maximum price on that item between the menus is the most updated one and if there are multiple menus with that condition the one with the maximum average price is the most updated one. Help him find the most updated menu.

In other words, a price on an item is good if it is the maximum price on that item among all menus, and a menu is the most updated one if it has the maximum number of good prices on it.  
If there are multiple menus with the maximum number of good prices, the menu with the higher price average is the most updated one.

**Input format**

* The first line contains integers n and m that denote the number of menus and the number of items on each menu respectively.
* The next n line each contains m integers represented as Ai,j, the jth price on the ith menu.

**Output format**

Print a single number denoting the number of the most updated menu.

It is guaranteed that the answer is unique.

**Constraints**

1≤n,m≤103

1≤Ai,j≤109

**Sample Input**

3 4

1 2 1 10

3 2 3 4

1 3 3 2

**Sample Output**

2

**Explanation**

There are 4 items in this example. The maximum price for the first three items is 3 and for the last item is 10.

First menu has only one good price which is the last one, Second menu has 2 good prices on it, which are first and third items and the last menu, has 2 good prices too.

Se between second and third menus, we have to compare averages. Average of second menu is 3+2+3+44=3 and average of third menu is 1+3+3+24=2.25, so the second menu would be the answer.

Your task is to construct a tower in N days by following these conditions:

* Every day you are provided with one disk of distinct size.
* The disk with larger sizes should be placed at the bottom of the tower.
* The disk with smaller sizes should be placed at the top of the tower.

The order in which tower must be constructed is as follows:

* You cannot put a new disk on the top of the tower until all the larger disks that are given to you get placed.

Print N lines denoting the disk sizes that can be put on the tower on the ith day.

**Input format**

* First line: N denoting the total number of disks that are given to you in the N subsequent days
* Second line: N integers in which the ith integers denote the size of the disks that are given to you on the ith day

**Note**: All the disk sizes are distinct integers in the range of  1 to N.

**Output format**

Print N lines. In the ith line, print the size of disks that can be placed on the top of the tower in descending order of the disk sizes.

If on the ithday no disks can be placed, then leave that line empty.

**Constraints**

1≤N≤106

1≤size of disk≤N

**Sample Input**

5

4 5 1 2 3

**Sample Output**

5 4

3 2 1

**Explanation**

On the first day, the disk of size **4**is given. But you cannot put the disk on the bottom of the tower as a disk of size **5** is still remaining.

On the second day, the disk of size **5** will be given so now disk of sizes **5** and **4** can be placed on the tower.

On the third and fourth day, disks cannot be placed on the tower as the disk of **3**needs to be given yet. Therefore, these lines are empty.

On the fifth day, all the disks of sizes **3**, **2**, and **1** can be placed on the top of the tower.