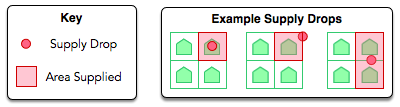
**Problem 1: Army Game**

Luke is daydreaming in Math class. He has a sheet of graph paper with “n” rows and “m” columns, and he imagines that there is an army base in each cell for a total of “n x m” bases. He wants to drop supplies at strategic points on the sheet, marking each drop point with a red dot. If a base contains at least one package inside or on top of its border fence, then it's considered to be supplied.

For example:



Given “n” and “m”, what's the minimum number of packages that Luke must drop to supply all of his bases?

**Input Format**

Two space-separated integers describing the respective values of “n” and “m”.

**Constraints**

0 < n <= 1000

0 < m <= 1000

**Output Format**

Print a single integer denoting the minimum number of supply packages Luke must drop.

***Find Sample Inputs/Outputs and explanations on next page***

**Sample Input**

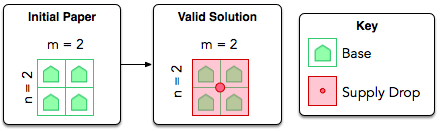
2 2

**Sample Output**

1

**Explanation**

Luke has four bases in a 2x2 grid. If he drops a single package where the walls of all four bases intersect, then those four cells can access the package:



Because he managed to supply all four bases with a single supply drop, we print 1 as our answer.

**Problem 2: Little Elephant and permutation**

The Little Elephant likes permutations. This time he has a permutation **A[1], A[2],…**,**A[N]** of numbers **1**, **2**,...,**N**.

He calls a permutation **A** good, if the number of its inversions is equal to the number of its local inversions. The number of inversions is defined as the number of pairs of integers (**i,j**) such that **1** ≤ **i** < **j** ≤ **N** and **A[i]** > **A[j]**, and the number of local inversions is the number of integers **i** such that **1** ≤ **i** < **N** and **A[i]** > **A[i+1]**.

The Little Elephant has several such permutations. Help him to find for each permutation whether it is good or not. Print **YES** for a corresponding test case if it is good and **NO** otherwise.

**Input Format**

The first line of the input contains a single integer **T**, the number of test cases. **T** test cases follow. The first line of each test case contains a single integer **N**, the size of a permutation. The next line contains **N** space separated integers **A[1]**, **A[2]**, ..., **A[N]**.

**Output Format**

For each test case output a single line containing the answer for the corresponding test case. It should be **YES** if the corresponding permutation is good and **NO** otherwise.

**Constraints**

**1** ≤ **T** ≤ **474**  
  
**1** ≤ **N** ≤ **100**  
  
It is guaranteed that the sequence **A[1]**, **A[2]**, ..., **A[N]** is a permutation of numbers **1**, **2**, ..., **N.**

***Find Example Input and Output on next page.***

**Input:**

4

1

1

2

2 1

3

3 2 1

4

1 3 2 4

**Output:**

YES

YES

NO

YES

**Explanation:**

**Case 1.** Here **N = 1**, so we have no pairs **(i, j)** with**1 ≤ i < j ≤ N**. So the number of inversions is equal to zero. The number of local inversion is also equal to zero. Hence this permutation is good.

**Case 2.** Here **N = 2**, and we have one pair **(i, j)** with**1 ≤ i < j ≤ N**, the pair**(1, 2)**. Since **A[1] = 2** and **A[2] = 1**,then **A[1] > A[2]** and the number of inversions is equal to**1**. The number of local inversion is also equal to **1** since we have one value of **i** for which**1 ≤ i < N** (the value**i = 1**) and **A[i] > A[i+1]** for this value of **i** since **A[1] > A[2]**. Hence this permutation is also good.

**Case 3.** Here **N = 3**, and we have three pairs **(i, j)** with **1 ≤ i < j ≤ N**. We have **A[1] = 3, A[2] = 2, A[3] = 1**. Hence **A[1] > A[2]**, **A[1] > A[3]** and **A[2] > A[3]**. So the number of inversions is equal to **3**. To count the number of local inversion we should examine inequalities **A[1] > A[2]** and **A[2] > A[3]**. They both are satisfied in our case, so we have **2** local inversions. Since **2 ≠ 3** this permutations is not good.

**Case 4.** Here we have only one inversion and it comes from the pair **(2, 3)** since **A[2] = 3 > 2 = A[3]**. This pair gives also the only local inversion in this permutation. Hence the number of inversions equals to the number of local inversions and equals to one. So this permutation is good.

**Problem 3: Matrix Tracing**

A word from the English dictionary is taken and arranged as a matrix. e.g. "MATHEMATICS"

MATHE

ATHEM

THEMA

HEMAT

EMATI

MATIC

ATICS

There are many ways to trace this matrix in a way that helps you construct this word. You start tracing the matrix from the top-left position and at each iteration, you either move RIGHT or DOWN, and ultimately reach the bottom-right of the matrix. It is assured that any such tracing generates the same word. How many such tracings can be possible for a given word of length “**m + n - 1**” written as a matrix of size **m** **x** **n**?

**Input Format**

The first line of input contains an integer T. T test cases follow.

Each test case contains 2 space separated integers **m** & **n** (in a new line) indicating that the matrix has **m** rows and each row has **n** characters.

**Constraints**

1 <= T <= 103

1 ≤ m ≤ 106

1 ≤ n ≤ 106

**Output Format**

Print the number of ways (call it **S**) the word can be traced as explained in the problem statement. If the number is larger than 10^9+7, print **S mod (10^9 + 7)** for each testcase (in a new line).

**Sample Input**

1

2 3

**Sample Output**

3

**Explanation**

Let's consider a word AWAY written as the matrix

AWA

WAY

Here, the word AWAY can be traced in 3 different ways, traversing either RIGHT or DOWN.

AWA

Y

AW

AY

A

WAY

Hence the answer is 3.

**Problem 4: Array Left Rotations**

A left rotation operation on an array of size “n” shifts each of the array's elements 1 unit to the left. For example, if 2 left rotations are performed on array [1,2,3,4,5], then the array would become [3,4,5,1,2].

Given an array of n integers and a number, k, perform k left rotations on the array. Then print the updated array as a single line of space-separated integers.

**Hint:** As the constraint for k is huge, think of a clever way to solve this problem efficiently, otherwise you will get a “time limit exceeded” judgement (worth 0 points).

Input Format

The first line contains two space-separated integers denoting the respective values of “n” (the number of integers) and “k” (the number of left rotations you must perform).  
The second line contains “n” space-separated integers describing the respective elements of the array's initial state.

Constraints

* 1 <= n <= 105
* 1 <= k <= 1018

Output Format

Print a single line of “n” space-separated integers denoting the final state of the array after performing “k” left rotations.

Sample Input

5 4

1 2 3 4 5

Sample Output

5 1 2 3 4

*Find explanation on next page*

Explanation

When we perform k = 4 left rotations, the array undergoes the following sequence of changes:

Original Array: [1, 2, 3, 4, 5]

1 Left Rotation: [2, 3, 4, 5, 1]

2 Left Rotations: [3, 4, 5, 1, 2]

3 Left Rotations: [4, 5, 1, 2, 3]

4 Left Rotations: [5, 1, 2, 3, 4]

Thus, we print the array's final state as a single line of space-separated values, which is 5 1 2 3 4.