**Problem1: 60Marks**

Alice likes to shop online. The main reason is that she gets a lot of discounts! There are **M** online shops from where she purchases her items. The prices of the same item vary in both the shops. She needs to buy **N** items in all. And each she can buy each item in any one of the shops. All the shops offer certain discounts. If she buys the *i*th item in one shop, she gets a discount coupon for the *i+1*th item in the same shop. But if she purchases the *i+1*th item from a different shop, she cannot use the discount coupon.  
For example, if she buys the *i*th item in Shop *j* and gets a discount of ***X*** rupees. Let the price of *i+1*th item be ***Y*** rupees in shop *j* and ***Z*** rupees in shop *k*. If she purchases the *i+1*th item from shop *j* she can use the discount coupon and hence, will have to pay ***Y-X***rupees. If she purchases the *i+1*th item from shop *k* she cannot use the discount coupon and hence, will have to pay ***Z*** rupees.  
  
**Note:** If the discount given is more than price of the item in the shop, i.e. if ***Y-X*** < 0, the item is given free but she does NOT get any money back.

Alice wants to spend as less as possible and wants your help. She starts buying from item 1 and goes on till item N in that order. Your task is simple. Tell her the least amount to spend to get all the **N** items.

**Input:**

First line of input contains a single integer **T**, the number of test cases.  
  
Each test case starts with a line consisting of two space separated integers **N** & **M**.  
The next **N** lines consist of **M** space separated integers each. The *j*th number on the *i*thline denotes the price of *i*th item in the *j*th shop.  
  
The next **N** lines consist of **M** space separated integers each. The *j*th number on the *i*thline denotes the value of discount coupon given after purchasing *i*th item in the *j*th shop.

**Output:**

For each test case, output on a separate line the least amount that has to be spent to get all the **N** items.

**Constraints:**

1 ≤ **T** ≤ 10

2 ≤ **N, M** ≤ 100000

0 ≤ Prices, Discounts ≤ 1000000

1 ≤ **T \* N \* M** ≤ 1000000

**Example:**

**Input:**

2

2 2

3 4

1 2

1 0

0 1

2 4

1 2 3 4

4 3 2 1

2 3 2 1

1 2 1 1

**Output:**

3

2

**Problem2: 60Marks**

As every other little boy, Mike has a favorite toy to play with. Mike's favorite toy is a set of **N** disks. The boy likes to compose his disks in stacks, but there's one very important rule: the disks in a single stack must be ordered by their radiuses in a **strictly** increasing order such that the top-most disk will have the smallest radius.

For example, a stack of disks with radii (5, 2, 1) is valid, while a stack of disks with radii (3, 4, 1) is not.

Little Mike has recently come up with the following algorithm after the order of disks are given:

* First, Mike initiates an empty set of disk stacks.
* Then, Mike processes the disks in the chosen order using the following pattern:
  + If there is at least one stack such that Mike can put the current disk on the top of the stack without making it invalid, then he chooses the stack with the smallest top disk radius **strictly** greater than the radius of the current disk, and puts the current disk on top of that stack.
  + Otherwise, Mike makes a new stack containing only the current disk.

For example, let's assume that the order of the disk radii is (3, 4, 5, 1, 1, 2). Here's how the set of the top stack disks will appear during the algorithm's run:

* In the beginning of the algorithm, the set of disk stacks is empty. After processing the first disk, the set of top stack disks is {3}.
* We cannot put the second disk on the only stack that we have after processing the first disk, so we make a new stack. After processing the second disk, the set of top stack disks is {3, 4}.
* We cannot put the third disk on any of the available stacks, so we make a new stack. After processing the third disk, the set of top stack disks is {3, 4, 5}.
* The fourth disk has radius 1, so it can be easily put on any of the available stacks. According to the algorithm, we choose the stack with the top disk radius equal to 3. After processing the fourth disk, the set of top stack disks is {1, 4, 5}.
* The fifth disk has radius 1, so there are two stacks we can put it on. According to the algorithm, we choose the stack with the top disk radius equal to 4. After processing the fifth disk, the set of top stack disks is {1, 1, 5}.
* The sixth disk has radius 2, so there is only one stack we can put it on. The final set of top stack disks is {1, 1, 2}.

Mike is really excited about his new algorithm, but he has so many disks that it seems impossible to simulate the algorithm manually.

You are given an array **A** of **N** integers denoting the radii of Mike's disks. The disks are already ordered by Mike. Your task is to find the set of the stack top disk radii after the algorithm is done.

### Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of a test description contains a single integer **N**.

The second line of the description contains **N** integers denoting **A1, ... , AN**.

### Output

For each test case, output a single line. The line should start with a positive integer **S**denoting the number of stacks after the algorithm is done. This should be followed by **S**integers on the same line denoting the stacks' top disk radii in **non-decreasing order**.

If there are multiple correct answers, you are allowed to output any of them.

### Constraints

* 1 ≤ **T** ≤ 10
* 1 ≤ **N** ≤ 105
* 1 ≤ **Ai** ≤ 109

### Example

**Input:**

3

6

3 4 5 1 1 2

10

3 2 9 5 2 9 4 14 7 10

8

14 5 13 19 17 10 18 12

**Output:**

3 1 1 2

5 2 2 4 7 10

4 5 10 12 18