Codeforces Round 1050 (Div. 4)

A. Sublime Sequence

1 second, 256 megabytes

Farmer John has an integer x. He creates a sequence of length n by alternating integers x and $-x$, starting with x.

For example, if $n = 5$, the sequence looks like: $x, -x, x, -x, x$.

He asks you to find the sum of all integers in the sequence.

Input

The first line contains an integer t ($1 \le t \le 100$) — the number of test cases.

The only line of input for each test case is two integers x and n ($1 \le x, n \le 10$).

Output

For each test case, output the sum of all integers in the sequence.

input

```
4
1 4
2 5
3 6
4 7
```

output

```
0
2
0
4
```

B. Lasers

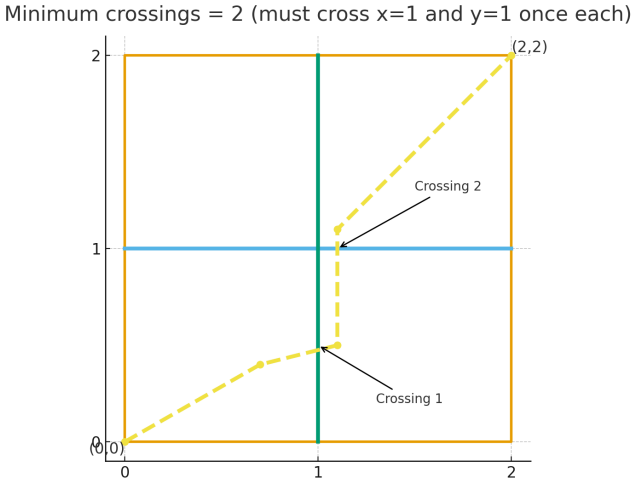
2 seconds, 256 megabytes

There is a 2D-coordinate plane that ranges from $(0, 0)$ to (x, y). You are located at $(0, 0)$ and want to head to (x, y).

However, there are n horizontal lasers, with the i-th laser continuously spanning $(0, a_i)$ to (x, a_i). Additionally, there are also m vertical lasers, with the i-th laser continuously spanning $(b_i, 0)$ to (b_i, y).

You may move in any direction to reach (x, y), but your movement must be a continuous curve that lies inside the plane. Every time you cross a vertical or a horizontal laser, it counts as one crossing. Particularly, if you pass through an intersection point between two lasers, it counts as **two crossings**.

For example, if $x = y = 2$, $n = m = 1$, $a = [1]$, $b = [1]$, the movement can be as follows:



What is the minimum number of crossings necessary to reach (x, y)?

Input

The first line contains t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains four integers n, m, x, and y ($1 \le n, m \le 2 \cdot 10^5, 2 \le x, y \le 10^9$).

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($0 < a_i < y$) — the y-coordinates of the horizontal lasers. It is guaranteed that $a_i > a_{i-1}$ for all $i > 1$.

The following line contains m integers $b_1, b_2, \ldots, b_m$ ($0 < b_i < x$) — the x-coordinates of the vertical lasers. It is guaranteed that $b_i > b_{i-1}$ for all $i > 1$.

It is guaranteed that the sum of n and m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the minimum number of crossings necessary to reach (x, y).

input

```
2
1 1 2 2
1
1
2 1 100000 100000
42 58
32
```

output

```
2
3
```

C. Pacer

2 seconds, 256 megabytes

The FitnessGram Pacer Test is a multistage aerobic capacity test that progressively gets more difficult as it continues. The 20 meter pacer test will begin in 30 seconds. Line up at the start. A single lap should be completed every time you hear this sound. Ding! Remember to run in a straight line and run as long as possible. The test will begin on the word start. On your mark. Get ready!...

Farmer John is running the FitnessGram Pacer Test! Farmer John takes **one minute** to run to the other side of the gym. Therefore, at the start of each minute, FJ can choose to either run to the other side of the gym or stay in place. If he chooses to run to the other side of the gym, he gains **one point**.

FJ will run the Pacer Test until the start of the m-th minute. Initially (at the start of the 0-th minute), FJ is at the starting side of the gym, which we will denote as side 0. The opposite side of the gym is denoted side 1.

The pacer test audio plays n times. At the start of the a_i-th minute, FJ must be at the b_i-th side of the gym.

What is the maximum number of points FJ can acquire while ensuring that he meets the audio's requirements?

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n \le 2 \cdot 10^5, n \le m \le 10^9$) — the number of requirements and the number of total minutes.

The following n lines contain two integers a_i and b_i ($1 \le a_i \le m, b_i \in \{0, 1\}$) — the i-th requirement by the audio. It is guaranteed that $a_i > a_{i-1}$ over all $i > 1$.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the maximum number of points that FJ can acquire.

input

```
3
2 4
2 1
4 0
2 7
1 1
4 0
4 9
1 0
2 0
6 1
9 0
```

output

```
2
7
6
```

For the first sample test case,

- During minute 0, FJ can stay at side 0.
- During minute 1, FJ can run to side 1 and gain 1 point.
- Right before minute 2, the audio requires FJ to be at side 1. Here, FJ is indeed at side 1.
- During minute 2, FJ can run to side 0 and gain 1 point.
- During minute 3, FJ can stay at side 0.
- Right before minute 4, the audio requires FJ to be at side 0. Here, FJ is indeed at side 0.
- Since the start of minute 4 has reached, the Pacer Test ends. His total score is 2.

Relevant illustration of the statement:

D. Destruction of the Dandelion Fields

2 seconds, 256 megabytes

Farmer John has a lawnmower, initially turned off. He also has n fields, with the i-th field having a_i dandelions. He will visit all the fields in any order he wants, and each field **exactly once**.

FJ's lawnmower seems to have a mind of its own. Right before visiting a field, it checks if the field has an even or odd number of dandelions. If it has an odd number, then the lawnmower toggles its state (if it is off, it turns on; if it is on, it turns off). Then, if the lawnmower is on, it will cut all dandelions in that field. Otherwise, if the lawnmower is off, then FJ will simply visit the field and cut no dandelions.

If FJ visits the n fields in optimal order, what is the maximum total number of dandelions he can cut?

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of fields.

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the number of dandelions in each field.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output an integer on a new line: maximum dandelions FJ can cut if he visits all n fields in optimal order.

input

```
3
3
2 4 6
4
4 2 1 6
4
1000000000 999999999 1000000000 999999999
```

output

```
0
13
2999999999
```

For the first test case, since there is no field with an odd number of dandelions, FJ can never turn his lawnmower on. Since his lawnmower is always off, he can never cut any dandelions, so the answer is 0.

For the second test case, FJ can visit the third field first; then his lawnmower will turn on. Then he can visit the other fields in any order. Since his lawnmower is always on, dandelions in every field can be cut.

For the third test case, FJ can visit the fields in the following order: field 2, field 1, field 3, then field 4.

E. Split

2 seconds, 256 megabytes

Farmer John has an array a containing n positive integers and an integer k.

Let $a[l, r]$ be a subarray* of a. He performs the following procedure to independently determine if subarray $a[l, r]$ is awesome:

- Initially, FJ has k empty multisets, numbered from 1 to k.
- Then, for each element a_i ($1 \le i \le n$) in a:
  - If $l \le i \le r$ (that is, a_i is in the subarray $a[l, r]$), he places a_i in multiset 1,
  - Otherwise, he places a_i into any multiset he wants **(which may be multiset 1)**.
- Subarray $a[l, r]$ is awesome if there is some way for him to place elements such that, for every value v, all multisets contain the same number of elements with value v. In other words, he wants to make all multisets contain the exact same elements (ignoring ordering).

Output the number of awesome subarrays.

*For array a of size n and integers $1 \le l \le r \le n$, the subarray $a[l, r]$ denotes the array consisting of the elements $a_l, \ldots, a_r$, in order.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains two integers n and k ($2 \le k \le n \le 2 \cdot 10^5$).

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each testcase, output one integer on a new line: the number of awesome subarrays.

input

```
4
3 2
1 1 1
4 2
1 2 1 2
8 2
3 3 3 2 2 2 2
6 3
1 1 1 1 1 1
```

output

```
0
7
18
11
```

Test case 1: $n = 3$, $a = [1, 1, 1]$.

For $k = 2$, you cannot finish with the same number of 1's in both multisets, so no subarray works.

Test case 2: $n = 4$, $a = [1, 2, 1, 2]$.

For $k = 2$, the final state must give each multiset exactly one 1 and one 2. Thus a valid subarray can contain at most one 1 and at most one 2.

F. Gravity Falls

2 seconds, 256 megabytes

Farmer John has n arrays $a_1, a_2, \ldots, a_n$ that may have different lengths. He will stack the arrays on top of each other, resulting in a grid with n rows. The arrays are left-aligned and can be stacked in any order he desires.

Then, gravity will take place. Any cell that is not on the bottom row and does not have an element beneath it will fall down a row. This process will be repeated until there are no cells that satisfy the aforementioned constraint.

Over all possible ways FJ can stack the arrays, output the lexicographically minimum bottom row after gravity takes place.

Input

The first line contains t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains n ($1 \le n \le 2 \cdot 10^5$).

For the following n lines, the first integer k_i ($1 \le k \le 2 \cdot 10^5$) denotes the length of a_i.

Then, k_i space-separated integers $a_{i_1}, a_{i_2}, \ldots, a_{i_{k_i}}$ follow ($1 \le a_{i_j} \le 2 \cdot 10^5$).

It is guaranteed that the sum of n and the sum of all k_i over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the lexicographically minimum bottom row on a new line.

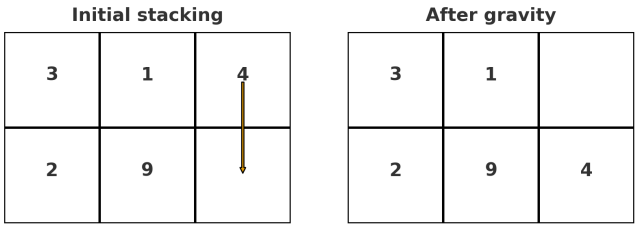
input

```
4
1
3 5 2 7
2
2 2 9
3 3 1 4
3
1 5
2 5 1
2 5 2
3
3 4 4 9
7 7 6 5 4 3 2 1
4 2 4 5 1
```

output

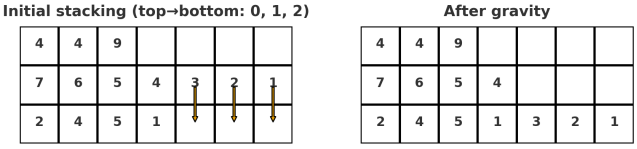
```
5 2 7
2 9 4
5 1
2 4 5 1 3 2 1
```

Demonstration for test case 2:



Bottom row: 2 9 4 (lexicographically minimum)

Demonstration for test case 4:



Bottom row: 2 4 5 1 3 2 1 (lexicographically minimum)

G. Farmer John's Last Wish

3 seconds, 256 megabytes

Bessie has found an array a of length n on the floor. There appears to be a handwritten note lying next to the array, seemingly written by Farmer John. The note reads:

Help me, dear Bessie! Let $f(a)$ denote the maximum integer k in the range $[1, n)$ such that $\gcd(a_1, a_2, \ldots, a_k) > \gcd(a_1, a_2, \ldots, a_{k+1})$, or 0 if no such k exists.

Bessie decides to help FJ. She defines $g(a)$ to represent the maximum value of $f(a)$ over all possible reorderings of a.

Bessie decides to not only find $g(a)$, but also the value of $g(p)$ for all prefixes p of a. Output n integers, the i'th of which is $g([a_1, a_2, \ldots, a_i])$.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$).

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers on a new line: the i'th of which should be $g([a_1, a_2, \ldots, a_i])$.

input

```
3
8
2 4 3 6 5 7 8 6
6
6 6 6 6 6 6
9
8 4 2 6 3 9 5 7 8
```

output

```
0 1 2 3 3 3 4 5
0 0 0 0 0 0
0 1 2 2 4 4 4 4 5
```