A. Equal Subsequences

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

We call a bitstring* perfect if it has the same number of `101` and `010` subsequences†. Construct a perfect bitstring of length n where the number of `1` characters it contains is exactly k.

It can be proven that the construction is always possible. If there are multiple solutions, output any of them.

---

*A bitstring is a string consisting only of the characters `0` and `1`.

†A sequence a is a subsequence of a string b if a can be obtained from b by the deletion of several (possibly zero or all) characters.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 100, 0 \le k \le n$) — the size of the bitstring and the number of `1` characters in the bitstring.

Output

For each test case, output the constructed bitstring. If there are multiple solutions, output any of them.

Standard Input	Standard Output
5 4 2 5 3 5 5 6 2 1 1	1010 10110 11111 100010 1

Note

In the first test case, the number of `101` and `010` subsequences is the same, both being 1, and the sequence contains exactly two `1` characters.

In the second test case, the number of `101` and `010` subsequences is the same, both being 2, and the sequence contains exactly three `1` characters.

In the third test case, the number of `101` and `010` subsequences is the same, both being 0, and the sequence contains exactly five `1` characters.

,

B. Make It Permutation

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

There is a matrix A of size $n \times n$ where $A_{i,j} = j$ for all $1 \le i, j \le n$.

In one operation, you can select a row and reverse any subarray* in it.

Find a sequence of at most $2n$ operations such that every column will contain a permutation† of length n.

It can be proven that the construction is always possible. If there are multiple solutions, output any of them.

---

*An array a is a subarray of an array b if a can be obtained from b by deleting zero or more elements from the beginning and zero or more elements from the end.

†A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2,3,1,5,4]$ is a permutation, but $[1,2,2]$ is not a permutation (2 appears twice in the array), and $[1,3,4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains one integer n ($3 \le n \le 5000$) — denoting the number of rows and columns in the matrix.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

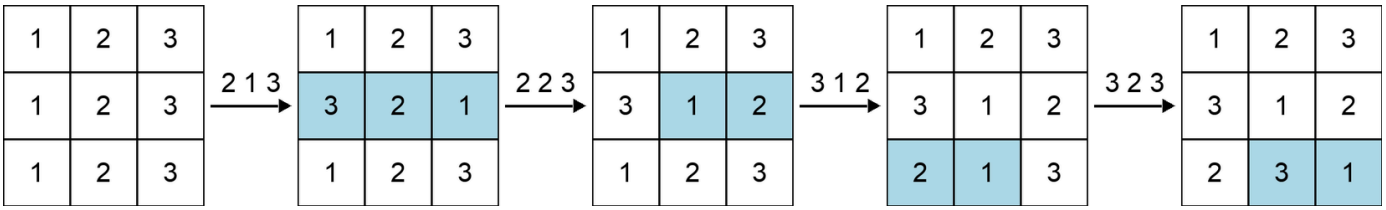
For each test case, on the first line, print an integer k ($0 \le k \le 2n$), the number of operations you wish to perform. On the next lines, you should print the operations.

To print an operation, use the format "$i\ l\ r$" ($1 \le l \le r \le n$ and $1 \le i \le n$) which reverses the subarray $A_{i,l}$, $A_{i,l+1}$, ..., $A_{i,r}$.

Standard Input	Standard Output
2 3 4	4 2 1 3 2 2 3 3 1 2 3 2 3 5 2 1 4 3 1 3 3 2 4 4 3 4 4 1 2

Note

In the first test case, the following operations are a valid solution:



,

C. Make It Beautiful

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

You are given an array a of n integers. We define the beauty of a number x to be the number of 1 bits in its binary representation. We define the beauty of an array to be the sum of beauties of the numbers it contains.

In one operation, you can select an index i $(1 \leq i \leq n)$ and increase a_i by 1.

Find the maximum beauty of the array after doing **at most** k operations.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 5000$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \leq n \leq 5000$, $0 \leq k \leq 10^{18}$) — the length of the array and the maximal number of operations.

The second line of each test case contains n integers $a_1, a_2, \ldots a_n$ ($0 \leq a_i \leq 10^9$) —denoting the array a.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, output a single integer, the maximum beauty after at most k operations.

Standard Input	Standard Output
5 5 2 0 1 7 2 4 5 3 0 1 7 2 4 1 1 3 3 0 2 0 3 1 100000000000 0	8 9 2 3 36

Note

In the first test case, $a = [0, 1, 7, 2, 4]$.

- apply the first operation at $i = 1$, the new array is $a = [1, 1, 7, 2, 4]$
- apply the second operation at $i = 4$, the new array is $a = [1, 1, 7, 3, 4]$

The beauty of this array is $1 + 1 + 3 + 2 + 1 = 8$. One of the other valid solutions with the same beauty is $[0, 1, 7, 3, 5]$.

In the third test case, $a = [3]$. Since you are not required to use exactly k operations, it is optimal to do none.

,

D1. Red Light, Green Light (Easy version)

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

This is the easy version of the problem. The only difference is the constraint on k and the total sum of n and q across all test cases. You can make hacks only if both versions of the problem are solved.

You are given a strip of length 10^{15} and a constant k. There are exactly n cells that contain a traffic light; each has a position p_i and an initial delay d_i for which $d_i < k$. The i-th traffic light works the following way:

- it shows red at the $l \cdot k + d_i$-th second, where l is an integer,
- it shows green otherwise.

At second 0, you are initially positioned at some cell on the strip, facing the positive direction. At each second, you perform the following actions in order:

- If the current cell contains a red traffic light, you turn around.
- Move one cell in the direction you are currently facing.

You are given q different starting positions. For each one, determine whether you will eventually leave the strip within 10^{100} seconds.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains two integers n, k ($\mathbf{1 \le n \le 500}$ and $\mathbf{1 \le k \le 500}$) — the number of traffic lights and the length of the period.

The second line of each test case contains n integers $p_1, p_2, \ldots p_n$ ($1 \le p_1 < p_2 < \cdots < p_n \le 10^{15}$) — the positions of the traffic lights.

The third line of each test case contains n integers $d_1, d_2, \ldots d_n$ ($0 \le d_i < k$) — the delays of the traffic lights.

The fourth line of each test case contains one integer q ($\mathbf{1 \le q \le 500}$) — the number of queries.

The fifth line of each test case contains q integers $a_1, a_2, \ldots, a_q$ ($1 \le a_i \le 10^{15}$) — the starting positions.

It is guaranteed that the sum of n and q over all test cases does not exceed 500.

Output

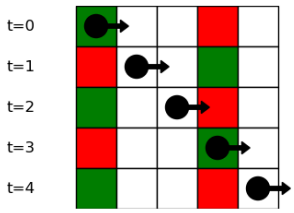
For each test case, output q lines. Each line should contain "YES" if you will eventually leave the strip and "NO" otherwise. You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

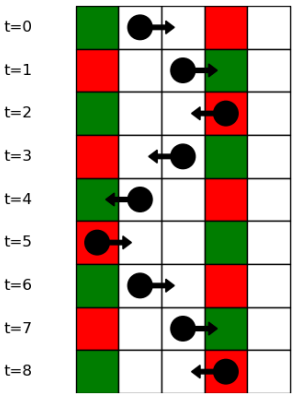
Standard Input	Standard Output
4 2 2 1 4 1 0	YES NO YES YES

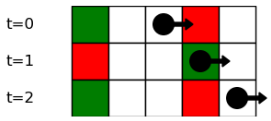
3 1 2 3 9 4 1 2 3 4 5 6 7 8 9 3 2 1 0 1 3 3 1 1 5 2 5 6 7 8 4 2 1 2 3 4 0 0 0 0 4 1 2 3 4 3 4 1 2 3 3 1 1 3 1 2 3	YES YES NO NO YES YES NO NO YES NO YES

Note

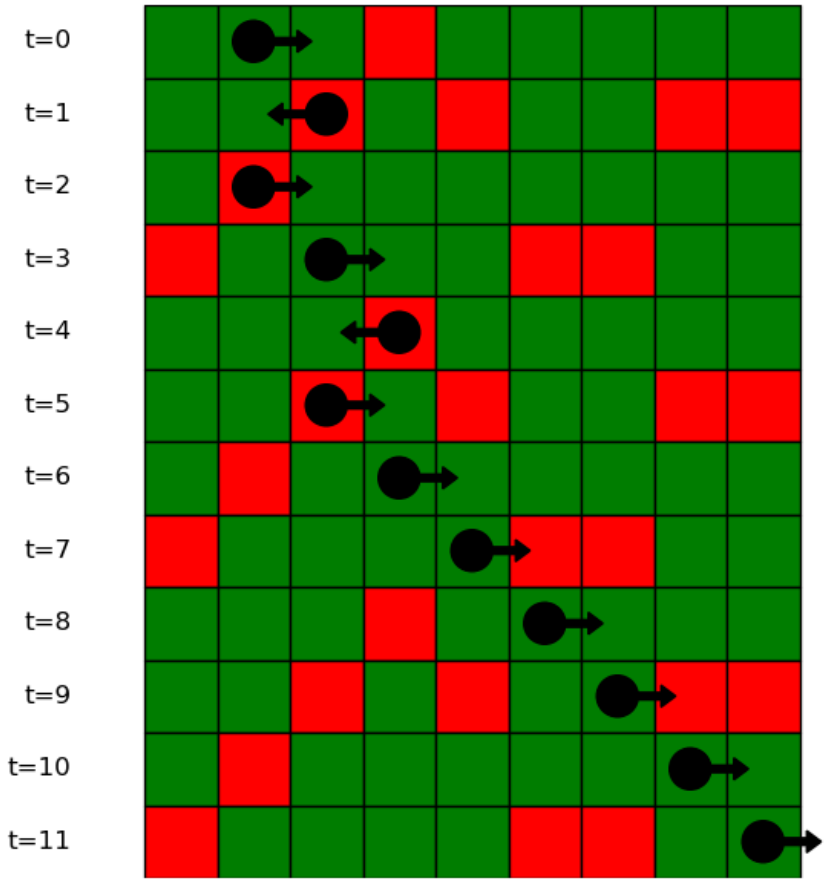
In the first test case, the following happens at starting positions 1, 2, and 3:







And the following in the second test case at starting position 2:


t=0
t=1
t=2
t=3
t=4
t=5
t=6
t=7
t=8
t=9
t=10
t=11

,

D2. Red Light, Green Light (Hard version)

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

This is the hard version of the problem. The only difference is the constraint on k and the total sum of n and q across all test cases. You can make hacks only if both versions of the problem are solved.

You are given a strip of length 10^{15} and a constant k. There are exactly n cells that contain a traffic light; each has a position p_i and an initial delay d_i for which $d_i < k$. The i-th traffic light works the following way:

- it shows red at the $l \cdot k + d_i$-th second, where l is an integer,
- it shows green otherwise.

At second 0, you are initially positioned at some cell on the strip, facing the positive direction. At each second, you perform the following actions in order:

- If the current cell contains a red traffic light, you turn around.
- Move one cell in the direction you are currently facing.

You are given q different starting positions. For each one, determine whether you will eventually leave the strip within 10^{100} seconds.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n, k ($\mathbf{1 \le n \le 2 \cdot 10^5}$ and $\mathbf{1 \le k \le 10^{15}}$) — the number of traffic lights and the length of the period.

The second line of each test case contains n integers $p_1, p_2, \ldots p_n$ ($1 \le p_1 < p_2 < \ldots < p_n \le 10^{15}$) — the positions of the traffic lights.

The third line of each test case contains n integers $d_1, d_2, \ldots d_n$ ($0 \le d_i < k$) — the delays of the traffic lights.

The fourth line of each test case contains one integer q ($\mathbf{1 \le q \le 2 \cdot 10^5}$) — the number of queries.

The fifth line of each test case contains q integers $a_1, a_2, \ldots, a_q$ ($1 \le a_i \le 10^{15}$) — the starting positions.

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

Output

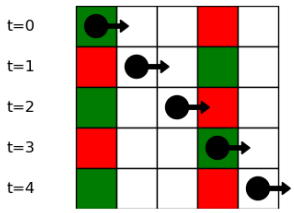
For each test case, output q lines. Each line should contain "YES" if you will eventually leave the strip and "NO" otherwise. You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

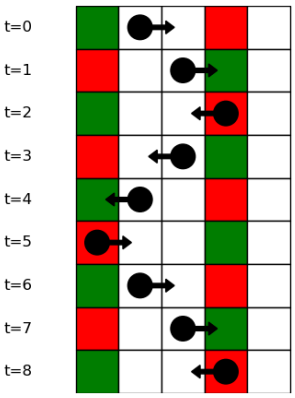
Standard Input	Standard Output
4 2 2 1 4 1 0	YES NO YES YES

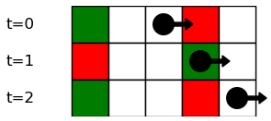
3 1 2 3 9 4 1 2 3 4 5 6 7 8 9 3 2 1 0 1 3 3 1 1 5 2 5 6 7 8 4 2 1 2 3 4 0 0 0 0 4 1 2 3 4 3 4 1 2 3 3 1 1 3 1 2 3	YES YES NO NO YES YES NO NO YES NO YES

Note

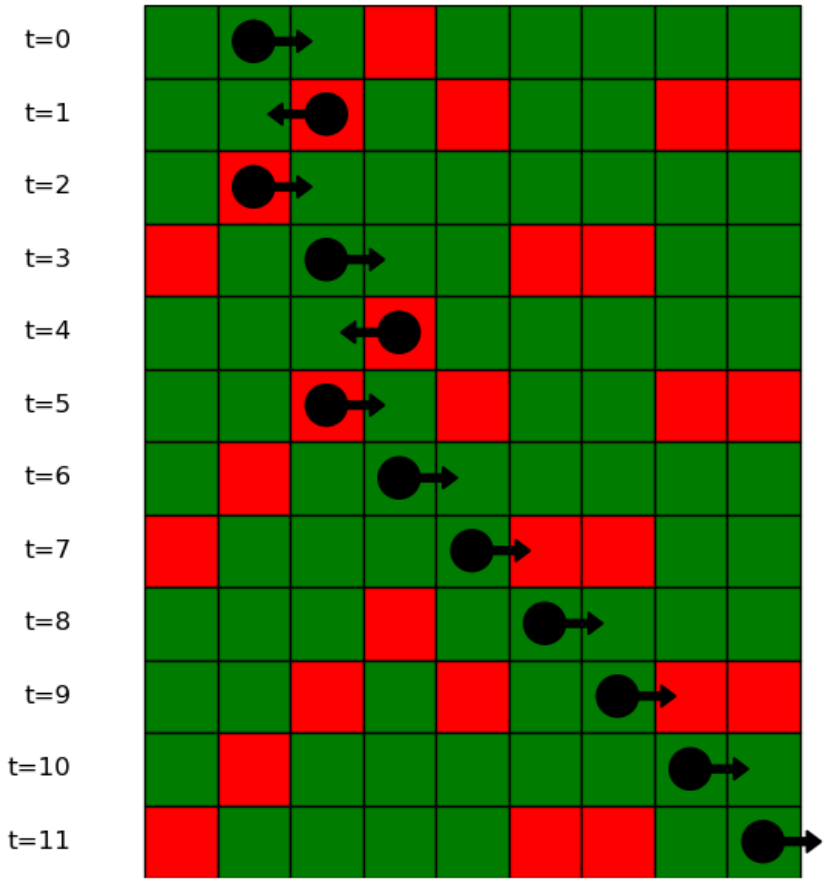
In the first test case, the following happens at starting positions 1, 2, and 3:







And the following in the second test case at starting position 2:


t=0
t=1
t=2
t=3
t=4
t=5
t=6
t=7
t=8
t=9
t=10
t=11

,

E. Grid Coloring

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

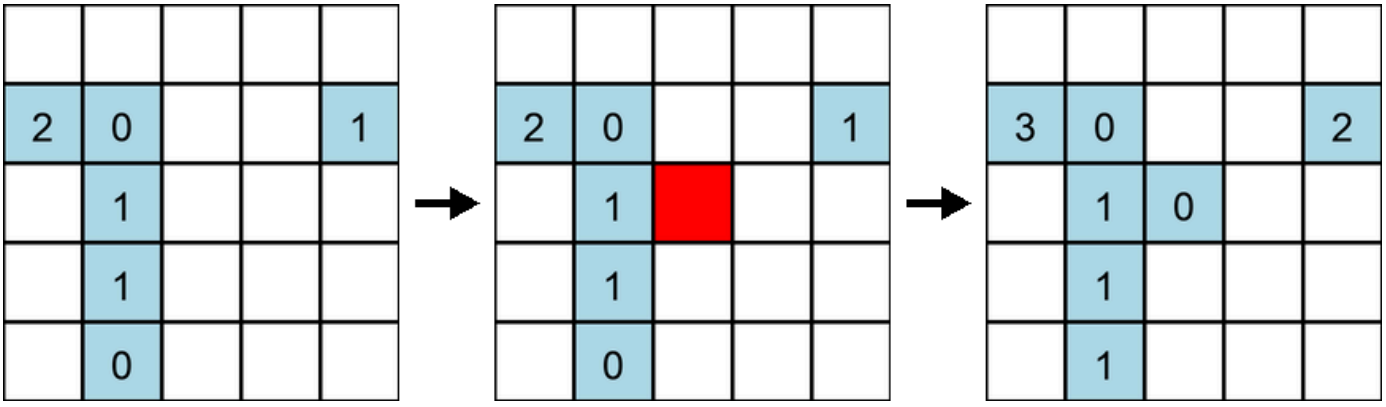
There is a $n \times m$ grid with each cell initially white. You have to color all the cells one-by-one. After you color a cell, all the **colored cells** furthest from it receive a penalty. Find a coloring order, where no cell has more than 3 penalties.

Note that n and m are both odd.

The distance metric used is the chessboard distance while we decide ties between cells with Manhattan distance. Formally, a cell (x_2, y_2) is further away than (x_3, y_3) from a cell (x_1, y_1) if one of the following holds:

- $\max\left(|x_1 - x_2|, |y_1 - y_2|\right) > \max\left(|x_1 - x_3|, |y_1 - y_3|\right)$
- $\max\left(|x_1 - x_2|, |y_1 - y_2|\right) = \max\left(|x_1 - x_3|, |y_1 - y_3|\right)$ **and** $|x_1 - x_2| + |y_1 - y_2| > |x_1 - x_3| + |y_1 - y_3|$

It can be proven that at least one solution always exists.



Example showing penalty changes after coloring the center of a 5×5 grid. The numbers indicate the penalty of the cells.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains two **odd** integers n and m ($1 \le n, m \le 4999$) — the number of rows and columns.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 5000.

Output

For each test case, output $n \cdot m$ lines where the i-th line should contain the coordinates of the i-th cell in your coloring order. If there are multiple solutions, print any of them.

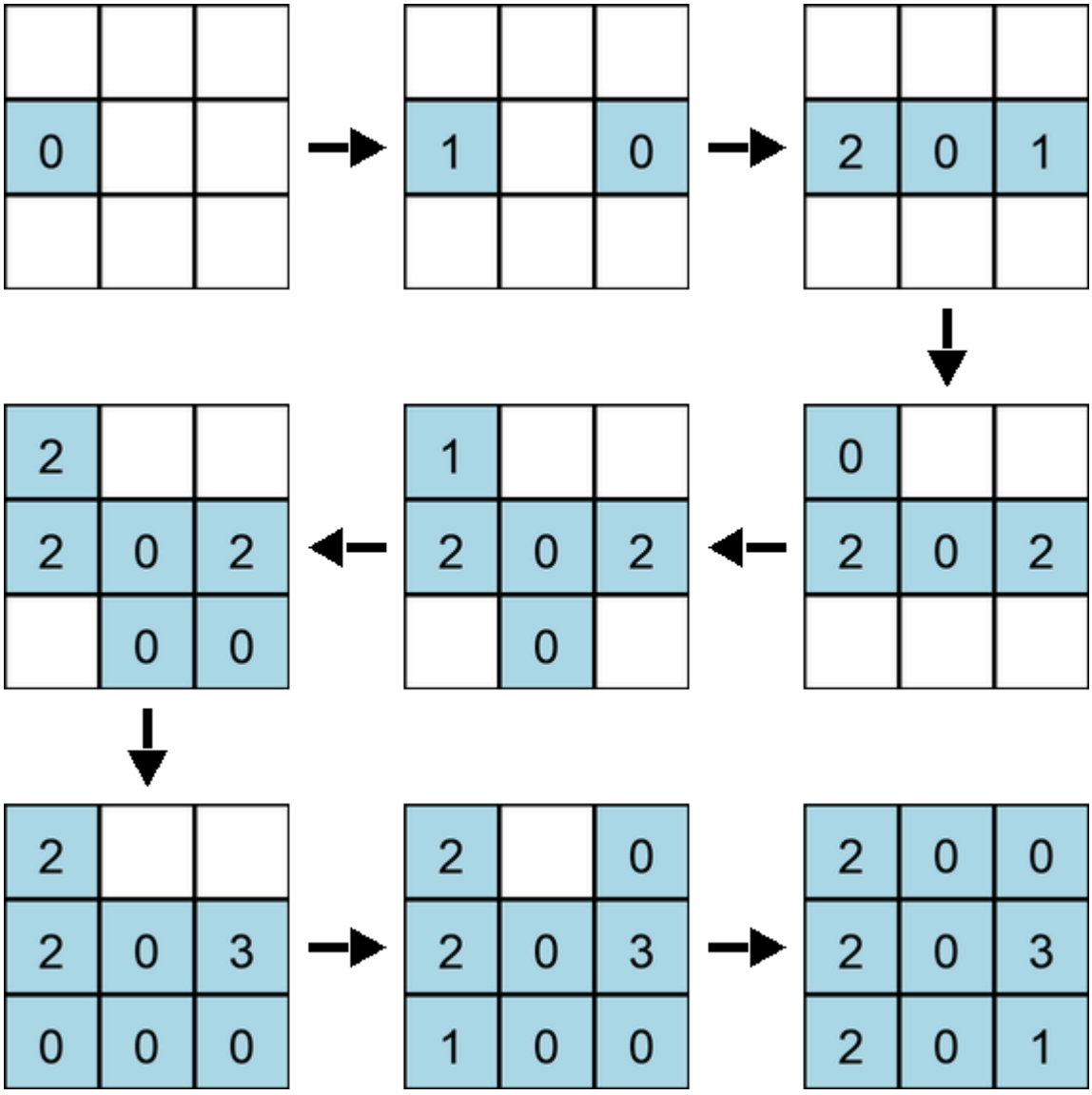
The empty lines in the example output are just for increased readability. You're not required to print them.

Standard Input	Standard Output
3 3 3 1 1	2 1 2 3 2 2

1 5	1 1 3 2 3 3 3 1 1 3 1 2 1 1 1 2 1 4 1 5 1 1 1 3

Note

In the first test case, the grid can be colored as follows:



The numbers indicate the penalty of the cells.

,

F. Shifts and Swaps

```
Input file:     standard input
Output file:    standard output
Time limit:     6 seconds
Memory limit:   512 megabytes
```

You are given arrays a and b of length n and an integer m.

The arrays only contain integers from 1 to m, and both arrays contain all integers from 1 to m.

You may repeatedly perform either of the following operations on a:

- cyclic shift* the array to the left
- swap two neighboring elements if their difference is at least 2.

Is it possible to transform the first array into the second?

---

*A left cyclic shift of a zero-indexed array p of length n is an array q such that $q_i = p_{(i+1) \bmod n}$ for all $0 \leq i < n$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \leq m \leq n \leq 5 \cdot 10^5$) — the length of the arrays and the number of distinct elements in a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq m$) — denoting the array a.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq m$) — denoting the array b.

It is guaranteed that both arrays contain all integers from 1 to m.

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

Output

For each test case, output "YES" if it is possible to transform the first array into the second and "NO" otherwise. You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
8 3 3 1 2 3 3 2 1 4 3 1 1 2 3 1 2 2 3 4 4 1 3 2 4 2 3 4 1 6 3 1 1 2 1 2 3	YES NO YES NO YES YES NO NO

2 1 1 2 3 1 5 4 2 3 4 1 1 3 2 1 1 4 9 7 2 4 6 7 3 1 5 4 6 6 7 3 5 6 4 2 4 1 9 8 8 3 5 6 5 4 1 7 2 7 5 3 5 8 4 6 2 1 8 6 2 1 5 4 6 3 5 4 6 1 5 2 4 5 3 4	

Note

In the first test case, you can transform array a into array b with the following steps:

- $[1, 2, 3]$ — shift to the left
- $[2, 3, 1]$ — swap indices 2 and 3
- $[2, 1, 3]$ — shift to the left
- $[1, 3, 2]$ — shift to the left
- $[3, 2, 1]$

In the second test case, it can be proven that it is impossible to transform array a into array b with the given operations.