Nikolas Mavrogeneiadis
gravitorious
August 21, 2024

Stefanos Set2

1. 1sec, 256MB

You received an $n \times m$ grid from a mysterious source. The source also gave you a magic positive integer constant k.

The source told you to color the grid with some colors, satisfying the following condition:

- If $(x_1, y_1), (x_2, y_2)$ are two distinct cells with the same color, then $\max(|x_1 - x_2|, |y_1 - y_2|) \geq k$.

You don't like using too many colors. Please find the minimum number of colors needed to color the grid.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The only line of each test case consists of three positive integers n, m, k ($1 \leq n, m, k \leq 10^4$) — the dimensions of the grid and the magic constant.

Output

For each test case, print a single integer — the minimum number of colors needed to color the grid.

Example

input Copy

```
6
3 3 2
5 1 10000
7 3 4
3 2 7
8 9 6
2 5 4
```

output Copy

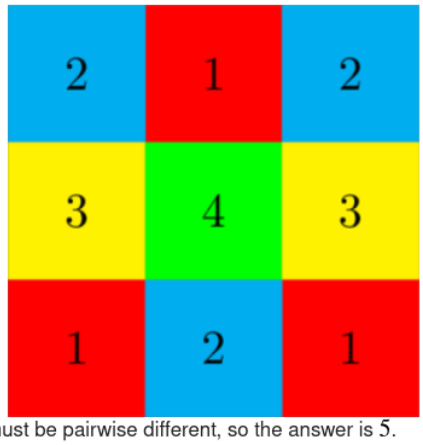
```
4
5
12
6
36
8
```

Figure 1:

1.

Note

In the first test case, one of the optimal constructions is:



In the second test case, the color of all cells must be pairwise different, so the answer is 5.

Figure 2:

2. **2secs, 256MB**

Suneet and Slavic play a card game. The rules of the game are as follows:

- Each card has an integer value between 1 and 10.
- Each player receives 2 cards which are face-down (so a player doesn't know their cards).
- The game is turn-based and consists **exactly of two turns**. In a round, both players pick a **random unflipped** card and flip it. The player who flipped a card with a strictly greater number wins the round. In case of equality, no one wins the round.
- A player wins a game if he wins the most number of rounds (i.e. strictly greater than the other player). In case of equality, no one wins the game.

Since Suneet and Slavic aren't best friends, you need to calculate the number of ways the game could happen that Suneet would end up as the winner.

For a better understanding, please check the notes section.

Input

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

The first and only line of each test case contains 4 integers a_1, a_2, b_1, b_2 ($1 \leq a_1, a_2, b_1, b_2 \leq 10$) where a_1 and a_2 represent the cards Suneet has, and b_1 and b_2 represent the cards Slavic has, respectively.

Output

For each test case, output a single integer — the number of games Suneet would win considering all possible games.

Figure 3:

2.

Example

input Copy

```
5
3 8 2 6
1 1 1 1
10 10 2 2
1 1 10 10
3 8 7 2
```

output Copy

```
2
0
4
0
2
```

Figure 4:

3. 1sec, 256MB

Alice got a permutation $a_1, a_2, \ldots, a_n$ of $[1, 2, \ldots, n]$, and Bob got another permutation $b_1, b_2, \ldots, b_n$ of $[1, 2, \ldots, n]$. They are going to play a game with these arrays.

In each turn, the following events happen in order:

- Alice chooses either the first or the last element of her array and removes it from the array;
- Bob chooses either the first or the last element of his array and removes it from the array.

The game continues for $n - 1$ turns, after which both arrays will have exactly one remaining element: x in the array a and y in the array b.

If $x = y$, Bob wins; otherwise, Alice wins. Find which player will win if both players play optimally.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 3 \cdot 10^5$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$, all a_i are distinct) — the permutation of Alice.

The next line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$, all b_i are distinct) — the permutation of Bob.

It is guaranteed that the sum of all n does not exceed $3 \cdot 10^5$.

Output

For each test case, print a single line with the name of the winner, assuming both players play optimally. If Alice wins, print `Alice`; otherwise, print `Bob`.

Figure 5:

3.

Example

input Copy

```
2
2
1 2
1 2
3
1 2 3
2 3 1
```

output Copy

```
Bob
Alice
```

Note

In the first test case, Bob can win the game by deleting the same element as Alice did.

In the second test case, Alice can delete 3 in the first turn, and then in the second turn, delete the element that is different from the one Bob deleted in the first turn to win the game.

Figure 6:

4. **1sec, 256MB**

You are given three integers x_c, y_c, and k ($-100 \le x_c, y_c \le 100$, $1 \le k \le 1000$).

You need to find k **distinct** points (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k), having integer coordinates, on the 2D coordinate plane such that:

- their center* is (x_c, y_c)
- $-10^9 \le x_i, y_i \le 10^9$ for all i from 1 to k

It can be proven that at least one set of k distinct points always exists that satisfies these conditions.

*The center of k points (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k) is $\left(\frac{x_1+x_2+\ldots+x_k}{k}, \frac{y_1+y_2+\ldots+y_k}{k}\right)$.

Input

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

Each test case contains three integers x_c, y_c, and k ($-100 \le x_c, y_c \le 100$, $1 \le k \le 1000$) — the coordinates of the center and the number of distinct points you must output.

It is guaranteed that the sum of k over all test cases does not exceed 1000.

Output

For each test case, output k lines, the i-th line containing two space separated integers, x_i and y_i, ($-10^9 \le x_i, y_i \le 10^9$) — denoting the position of the i-th point.

If there are multiple answers, print any of them. It can be shown that a solution always exists under the given constraints.

Figure 7:

4.

Example

input Copy

```
4
10 10 1
0 0 3
-5 -8 8
4 -5 3
```

output Copy

```
10 10
-1 -1
5 -1
-4 2
-6 -7
-5 -7
-4 -7
-4 -8
-4 -9
-5 -9
-6 -9
-6 -8
1000 -1000
-996 995
8 -10
```

Note

For the first test case, $\left(\frac{10}{1}, \frac{10}{1}\right) = (10, 10)$.

For the second test case, $\left(\frac{-1+5-4}{3}, \frac{-1-1+2}{3}\right) = (0, 0)$.

Figure 8:

5. 2secs, 256MB

In Berland, a bus consists of a row of n seats numbered from 1 to n. Passengers are advised to always board the bus following these rules:

- If there are no occupied seats in the bus, a passenger can sit in any free seat;
- Otherwise, a passenger should sit in any free seat that has at least one occupied neighboring seat. In other words, a passenger can sit in a seat with index i ($1 \le i \le n$) only if at least one of the seats with indices $i - 1$ or $i + 1$ is occupied.

Today, n passengers boarded the bus. The array a chronologically records the seat numbers they occupied. That is, a_1 contains the seat number where the first passenger sat, a_2 — the seat number where the second passenger sat, and so on.

You know the contents of the array a. Determine whether all passengers followed the recommendations.

For example, if $n = 5$, and $a = [5, 4, 2, 1, 3]$, then the recommendations were not followed, as the 3-rd passenger sat in seat number 2, while the neighboring seats with numbers 1 and 3 were free.

Input

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

The following describes the input test cases.

The first line of each test case contains exactly one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of seats in the bus and the number of passengers who boarded the bus.

The second line of each test case contains n **distinct** integers a_i ($1 \le a_i \le n$) — the seats that the passengers occupied in chronological order.

Figure 9:

5.

Output

For each test case, output on a separate line:

- "YES", if all passengers followed the recommendations;
- "NO" otherwise.

You may output the answer in any case (for example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as a positive answer).

Example

input Copy

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

output Copy

```
NO
YES
YES
NO
```

Figure 10:

6. **2secs, 256MB**

Slavic has a very tough exam and needs your help in order to pass it. Here is the question he is struggling with:

There exists a string s, which consists of lowercase English letters and possibly zero or more "?".

Slavic is asked to change each "?" to a lowercase English letter such that string t becomes a subsequence (not necessarily continuous) of the string s.

Output any such string, or say that it is impossible in case no string that respects the conditions exists.

Input

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

The first line of each test case contains a single string s ($1 \le |s| \le 2 \cdot 10^5$, and s consists only of lowercase English letters and "?"-s) – the original string you have.

The second line of each test case contains a single string t ($1 \le |t| \le |s|$, and t consists only of lowercase English letters) – the string that should be a subsequence of string s.

The sum of $|s|$ over all test cases doesn't exceed $2 \cdot 10^5$, where $|x|$ denotes the length of the string x.

Figure 11:

6.

Output

For each test case, if no such string exists as described in the statement, output "NO" (without quotes).

Otherwise, output "YES" (without quotes). Then, output one line — the string that respects all conditions.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes", and "Yes" will be recognized as a positive response).

If multiple answers are possible, you can output any of them.

Example

input Copy

```
5
?????
xbx
ab??e
abcde
ayy?x
a
ab??e
dac
paiu
mom
```

output Copy

```
YES
xabax
YES
abcde
YES
ayyyx
NO
NO
```

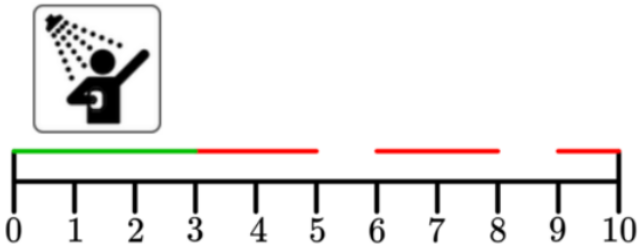
Figure 12:

7. 2secs, 256MB

As a computer science student, Alex faces a hard challenge — showering. He tries to shower daily, but despite his best efforts there are always challenges. He takes s minutes to shower and a day only has m minutes!

He already has n tasks planned for the day. Task i is represented as an interval (l_i, r_i), which means that Alex is busy and can not take a shower in that time interval (at any point in time strictly between l_i and r_i). **No two tasks overlap.**

Given all n time intervals, will Alex be able to shower that day? In other words, will Alex have a free time interval of length at least s?



In the first test case, Alex can shower for the first 3 minutes of the day and not miss any of the tasks.

Input

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

The first line of each test case contains three integers n, s, and m ($1 \le n \le 2 \cdot 10^5$; $1 \le s, m \le 10^9$) — the number of time intervals Alex already has planned, the amount of time Alex takes to take a shower, and the amount of minutes a day has.

Then n lines follow, the i-th of which contains two integers l_i and r_i ($0 \le l_i < r_i \le m$) — the time interval of the i-th task. No two tasks overlap.

Additional constraint on the input: $l_i > r_{i-1}$ for every $i > 1$.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Figure 13:

7.

Output

For each test case output "YES" (without quotes) if Alex can take a shower for that given test case, and "NO" (also without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes", and "Yes" will be recognized as a positive response).

Example

input Copy

```
4
3 3 10
3 5
6 8
9 10
3 3 10
1 2
3 5
6 7
3 3 10
1 2
3 5
6 8
3 4 10
1 2
6 7
8 9
```

output Copy

```
YES
YES
NO
YES
```

Figure 14:

8. 2secs, 256MB

You are given a cyclic array $a_1, a_2, \ldots, a_n$.

You can perform the following operation on a at most $n - 1$ times:

- Let m be the current size of a, you can choose any two adjacent elements where the previous one is no greater than the latter one (In particular, a_m and a_1 are adjacent and a_m is the previous one), and delete exactly one of them. In other words, choose an integer i ($1 \le i \le m$) where $a_i \le a_{(i \bmod m)+1}$ holds, and delete exactly one of a_i or $a_{(i \bmod m)+1}$ from a.

Your goal is to find the minimum number of operations needed to make all elements in a equal.

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 a single integer n ($1 \le n \le 100$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of array a.

Output

For each test case, output a single line containing an integer: the minimum number of operations needed to make all elements in a equal.

Figure 15:

8.

input Copy

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

output Copy

```
0
2
1
4
4
6
3
```

Note

In the first test case, there is only one element in a, so we can't do any operation.

In the second test case, we can perform the following operations to make all elements in a equal:

- choose $i = 2$, delete a_3, then a would become $[1, 2]$.
- choose $i = 1$, delete a_1, then a would become $[2]$.

It can be proven that we can't make all elements in a equal using fewer than 2 operations, so the answer is 2.

Figure 16:

9. 2secs, 256MB

There is an integer sequence a of length n, where each element is initially -1.

Misuki has two typewriters where the first one writes letters from left to right, with a pointer initially pointing to 1, and another writes letters from right to left with a pointer initially pointing to n.

Misuki would choose one of the typewriters and use it to perform the following operations until a becomes a permutation of $[1, 2, \ldots, n]$

- write number: write the minimum **positive** integer that isn't present in the array a to the element a_i, i is the position where the pointer points at. Such operation can be performed only when $a_i = -1$.
- carriage return: return the pointer to its initial position (i.e. 1 for the first typewriter, n for the second)
- move pointer: move the pointer to the next position, let i be the position the pointer points at before this operation, if Misuki is using the first typewriter, $i := i + 1$ would happen, and $i := i - 1$ otherwise. Such operation can be performed only if after the operation, $1 \le i \le n$ holds.

Your task is to construct any permutation p of length n, such that the minimum number of carriage return operations needed to make $a = p$ is the same no matter which typewriter Misuki is using.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the permutation.

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

Figure 17:

9.

Output

For each test case, output a line of n integers, representing the permutation p of length n such that the minimum number of carriage return operations needed to make $a = p$ is the same no matter which typewriter Misuki is using, or -1 if it is impossible to do so.

If there are multiple valid permutations, you can output any of them.

Example

input Copy

```
3
1
2
3
```

output Copy

```
1
-1
3 1 2
```

Figure 18:

9.

Note

In the first testcase, it's possible to make $a = p = [1]$ using 0 carriage return operations.

In the second testcase, it is possible to make $a = p = [1, 2]$ with the minimal number of carriage returns as follows:

If Misuki is using the first typewriter:

- Write number: write 1 to a_1, a becomes $[1, -1]$
- Move pointer: move the pointer to the next position. (i.e. 2)
- Write number: write 2 to a_2, a becomes $[1, 2]$

If Misuki is using the second typewriter:

- Move pointer: move the pointer to the next position. (i.e. 1)
- Write number: write 1 to a_1, a becomes $[1, -1]$
- Carriage return: return the pointer to 2.
- Write number: write 2 to a_2, a becomes $[1, 2]$

It can be proven that the minimum number of carriage returns needed to transform a into p when using the first typewriter is 0 and it is 1 when using the second one, so this permutation is not valid.

Similarly, $p = [2, 1]$ is also not valid, so there is no solution for $n = 2$.

In the third testcase, it is possibile to make $a = p = [3, 1, 2]$ with 1 carriage return with both the first and the second typewriter. It can be proven that, for both typewriters, it is impossible to write p with 0 carriage returns.

With the first typewriter it is possible to:

- Move pointer: move the pointer to the next position. (i.e. 2)
- Write number: write 1 to a_2, a becomes $[-1, 1, -1]$
- Move pointer: move the pointer to the next position. (i.e. 3)
- Write number: write 2 to a_3, a becomes $[-1, 1, 2]$
- Carriage return: return the pointer to 1.
- Write number: write 3 to a_1, a becomes $[3, 1, 2]$

With the second typewriter it is possible to:

- Move pointer: move the pointer to the next position. (i.e. 2)
- Write number: write 1 to a_2, a becomes $[-1, 1, -1]$
- Carriage return: return the pointer to 3.
- Write number: write 2 to a_3, a becomes $[-1, 1, 2]$
- Move pointer: move the pointer to the next position. (i.e. 2)
- Move pointer: move the pointer to the next position. (i.e. 1)
- Write number: write 3 to a_1, a becomes $[3, 1, 2]$

Figure 19:

10. CSES1 - Introductory Problems

11. CSES2 - Introductory Problems

12. CSES3 - Introductory Problems

13. CSES4 - Introductory Problems

14. CSES5 - Introductory Problems