Codeforces Round 972 (Div. 2)

A. Simple Palindrome

1 second, 256 megabytes

Narek has to spend 2 hours with some 2-year-old kids at the kindergarten. He wants to teach them competitive programming, and their first lesson is about palindromes.

Narek found out that the kids only know the vowels of the English alphabet (the letters `a`, `e`, `i`, `o`, and `u`), so Narek needs to make a string that consists of vowels only. After making the string, he'll ask the kids to count the number of subsequences that are palindromes. Narek wants to keep it simple, so he's looking for a string such that the amount of palindrome subsequences is minimal.

Help Narek find a string of length n, consisting of **lowercase** English **vowels only** (letters `a`, `e`, `i`, `o`, and `u`), which **minimizes** the amount of **palindrome**[†] **subsequences**[‡] in it.

[†] A string is called a palindrome if it reads the same from left to right and from right to left.

[‡] String t is a subsequence of string s if t can be obtained from s by removing several (possibly, zero or all) characters from s and concatenating the remaining ones, without changing their order. For example, `odocs` is a subsequence of `codeforces`.

Input

The first line of the input contains a single integer t $(1 \le t \le 100)$ — the number of test cases. Subsequently, the description of each test case follows.

The only line of each test case contains a single integer n ($1 \le n \le 100$) — the size of the string.

Output

For each test case, output any string of length n that satisfies the above conditions.

input
```
3
2
3
6
```
output
```
uo
iae
oeiiua
```

In the first example, `uo` has only three palindrome subsequences: `u`, `o`, and the empty string. It can be shown that there is no better answer.

In the third example, `oeiiua` has only eight palindrome subsequences: `o`, `e`, `i`, `i`, `u`, `a`, `ii`, and the empty string. It can be shown that there is no better answer.

B1. The Strict Teacher (Easy Version)

1.5 seconds, 256 megabytes

This is the easy version of the problem. The only differences between the two versions are the constraints on m and q. In this version, $m = 2$ and $q = 1$. You can make hacks only if both versions of the problem are solved.

Narek and Tsovak were busy preparing this round, so they have not managed to do their homework and decided to steal David's homework. Their strict teacher noticed that David has no homework and now wants to punish him. She hires other teachers to help her catch David. And now m teachers together are chasing him. Luckily, the classroom is big, so David has many places to hide.

The classroom can be represented as a one-dimensional line with cells from 1 to n, inclusive.

At the start, all m teachers and David are in **distinct** cells. Then they make moves. During each move

- David goes to an adjacent cell or stays at the current one.
- Then, each of the m teachers simultaneously goes to an adjacent cell or stays at the current one.

This continues until David is caught. David is caught if any of the teachers (possibly more than one) is located in the same cell as David. **Everyone sees others' moves, so they all act optimally.**

Your task is to find how many moves it will take for the teachers to catch David if they all act optimally.

Acting optimally means the student makes his moves in a way that maximizes the number of moves the teachers need to catch him; and the teachers coordinate with each other to make their moves in a way that minimizes the number of moves they need to catch the student.

Also, as Narek and Tsovak think this task is easy, they decided to give you q queries on David's position. **Note:** this is the easy version, and you are given **only one query**.

Input

In the first line of the input, you are given a single integer t ($1 \le t \le 10^5$) — the number of test cases. The description of each test case follows.

In the first line of each test case, you are given three integers n, m, and q $(3 \le n \le 10^9, m = 2, q = 1)$ — the number of cells on the line, the number of teachers, and the number of queries.

In the second line of each test case, you are given m distinct integers $b_1, b_2, \ldots, b_m$ $(1 \le b_i \le n)$ — the cell numbers of the teachers.

In the third line of each test case, you are given q integers $a_1, a_2, \ldots, a_q$ $(1 \le a_i \le n)$ — David's cell number for every query.

It is guaranteed that for any i, j such that $1 \le i \le m$ and $1 \le j \le q$, $b_i \neq a_j$.

Output

For each test case, output q lines, the i-th of them containing the answer of the i-th query.

input
```
3
10 2 1
1 4
2
8 2 1
3 6
1
8 2 1
3 6
8
```

output

```
1
2
2
```

In the first example, the student can just stay at cell 2. The teacher, initially located in cell 1, can reach cell 2 in one move. Therefore, the answer is 1.

In the second example, the student should just stay at cell 1. The teacher, initially located in cell 3, can reach cell 1 in two moves. Therefore, the answer is 2.

B2. The Strict Teacher (Hard Version)

1.5 seconds, 256 megabytes

This is the hard version of the problem. The only differences between the two versions are the constraints on m and q. In this version, $m, q \le 10^5$. You can make hacks only if both versions of the problem are solved.

Narek and Tsovak were busy preparing this round, so they have not managed to do their homework and decided to steal David's homework. Their strict teacher noticed that David has no homework and now wants to punish him. She hires other teachers to help her catch David. And now m teachers together are chasing him. Luckily, the classroom is big, so David has many places to hide.

The classroom can be represented as a one-dimensional line with cells from 1 to n, inclusive.

At the start, all m teachers and David are in **distinct** cells. Then they make moves. During each move

- David goes to an adjacent cell or stays at the current one.
- Then, each of the m teachers simultaneously goes to an adjacent cell or stays at the current one.

This continues until David is caught. David is caught if any of the teachers (possibly more than one) is located in the same cell as David. **Everyone sees others' moves, so they all act optimally.**

Your task is to find how many moves it will take for the teachers to catch David if they all act optimally.

Acting optimally means the student makes his moves in a way that maximizes the number of moves the teachers need to catch him; and the teachers coordinate with each other to make their moves in a way that minimizes the number of moves they need to catch the student.

Also, as Narek and Tsovak think this task is easy, they decided to give you q queries on David's position.

Input

In the first line of the input, you are given a single integer t ($1 \le t \le 10^5$) — the number of test cases. The description of each test case follows.

In the first line of each test case, you are given three integers n, m, and q ($3 \le n \le 10^9$, $1 \le m, q \le 10^5$) — the number of cells on the line, the number of teachers, and the number of queries.

In the second line of each test case, you are given m distinct integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le n$) — the cell numbers of the teachers.

In the third line of each test case, you are given q integers $a_1, a_2, \ldots, a_q$ ($1 \le a_i \le n$) — David's cell number for every query.

It is guaranteed that for any i, j such that $1 \le i \le m$ and $1 \le j \le q$, $b_i \ne a_j$.

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

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

Output

For each test case, output q lines, the i-th of them containing the answer of the i-th query.

input

```
2
8 1 1
6
3
10 3 3
1 4 8
2 3 10
```

output

```
5
1
1
2
```

In the only query of the first example, the student can run to cell 1. It will take the teacher five moves to reach from cell 6 to cell 1, so the answer is 5.

In the second query of the second example, the student can just stay at cell 3. The teacher, initially located in cell 4, can reach cell 3 in one move. Therefore, the answer is 1.

C. Lazy Narek

2 seconds, 256 megabytes

Narek is too lazy to create the third problem of this contest. His friend Artur suggests that he should use ChatGPT. ChatGPT creates n problems, each consisting of m letters, so Narek has n strings. To make the problem harder, he combines the problems by selecting some of the n strings **possibly none** and concatenating them **without altering their order**. His chance of solving the problem is defined as $score_n - score_c$, where $score_n$ is Narek's score and $score_c$ is ChatGPT's score.

Narek calculates $score_n$ by examining the selected string (he moves from left to right). He initially searches for the letter **"n"**, followed by **"a"**, **"r"**, **"e"**, and **"k"**. Upon finding all occurrences of these letters, he increments $score_n$ by 5 and resumes searching for **"n"** again (he doesn't go back, and he just continues from where he left off).

After Narek finishes, ChatGPT scans through the array and increments $score_c$ by 1 for each letter **"n"**, **"a"**, **"r"**, **"e"**, or **"k"** that Narek fails to utilize (note that if Narek fails to complete the last occurrence by finding all of the 5 letters, then all of the letters he used are counted in ChatGPT's score $score_c$, and Narek doesn't get any points if he doesn't finish finding all the 5 letters).

Narek aims to maximize the value of $score_n - score_c$ by selecting the most optimal subset of the initial strings.

Input

In the first line of the input, you're given a single integer t ($1 \le t \le 10^5$), the number of test cases. Then the description of each test case follows.

In the first line of each test case, you're given two integers n, m ($1 \le n, m \le 10^3$), the number of strings and the length of each string.

In the next n lines, you're given n strings, each having a length of m. The strings only contain lowercase letters of the English alphabet.

The sum of values of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case, output a single integer: the maximal possible value of $score_n - score_c$.

input

```
4
5 2
nn
aa
rr
ee
kk
1 5
narek
1 4
nare
5 7
nrrarek
nrnekan
uuuuuuu
ppppppp
nkarekz
```

output

```
0
5
0
7
```

In the first test case, one of the optimal answers is when Narek doesn't choose any of the strings, so the answer is 0. He can alternatively choose all the strings. In this case, the full string becomes "nnaarreekk". Narek can choose the first appearances of all letters and add 5 to the score. His opponent will add 1 for all second appearances, which will be 5 in total. So the answer will be $5 - 5 = 0$.

In the third test case, the only optimal answer is when Narek doesn't choose the string. Note that if he were to choose the string, he wouldn't be able to find the last letter "k", so his score would stay at 0 instead of becoming 5. Then ChatGPT would add 4 for all of the 4 letters, and the answer would become $0 - 4 = -4$.

In the last test case, Narek needs to choose the first and the last strings. After putting these two next to each other, he gets "$nrrareknkarekz$". Narek can choose the letters marked with red and add 10 to his score. Since the black colored letters Narek left behind are eligible for the opponent to claim (they are used in the word "narek"), the opponent adds all other letters to the score and gets a score of 3. Therefore, the answer is $10 - 3 = 7$.

D. Alter the GCD

4 seconds, 256 megabytes

You are given two arrays $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$.

You must perform the following operation **exactly once**:

- choose any indices l and r such that $1 \le l \le r \le n$;
- swap a_i and b_i for all i such that $l \le i \le r$.

Find the maximum possible value of $\gcd(a_1, a_2, \ldots, a_n) + \gcd(b_1, b_2, \ldots, b_n)$ after performing the operation exactly once. Also find the number of distinct pairs (l, r) which achieve the maximum value.

Input

In the first line of the input, you are given a single integer t ($1 \le t \le 10^5$), the number of test cases. Then the description of each test case follows.

In the first line of each test case, you are given a single integer n ($1 \le n \le 2 \cdot 10^5$), representing the number of integers in each array.

In the next line, you are given n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array a.

In the last line, you are given n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the elements of the array b.

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

Output

For each test case, output a line with two integers: the maximum value of $\gcd(a_1, a_2, \ldots, a_n) + \gcd(b_1, b_2, \ldots, b_n)$ after performing the operation exactly once, and the number of ways.

input

```
5
8
11 4 16 17 3 24 25 8
8 10 4 21 17 18 25 21
4
6 4 24 13
15 3 1 14
2
13 14
5 8
8
20 17 15 11 21 10 3 7
9 9 4 20 14 9 13 1
2
18 13
15 20
```

output

```
2 36
3 2
2 3
2 36
6 1
```

In the first, third, and fourth test cases, there's no way to achieve a higher GCD than 1 in any of the arrays, so the answer is $1 + 1 = 2$. Any pair (l, r) achieves the same result; for example, in the first test case there are 36 such pairs.

In the last test case, you must choose $l = 1, r = 2$ to maximize the answer. In this case, the GCD of the first array is 5, and the GCD of the second array is 1, so the answer is $5 + 1 = 6$, and the number of ways is 1.

E1. Subtangle Game (Easy Version)

2 seconds, 256 megabytes

This is the easy version of the problem. The differences between the two versions are the constraints on all the variables. You can make hacks only if both versions of the problem are solved.

Tsovak and Narek are playing a game. They have an array a and a matrix b of integers with n rows and m columns, numbered from 1. The cell in the i-th row and the j-th column is (i, j).

They are looking for the elements of a in turns; Tsovak starts first. Each time a player looks for a cell in the matrix containing the current element of a (Tsovak looks for the first, then Narek looks for the second, etc.). Let's say a player has chosen the cell (r, c). The next player has to choose his cell in the submatrix starting at $(r + 1, c + 1)$ and ending in (n, m) (the submatrix can be empty if $r = n$ or $c = m$). If a player cannot find such a cell (or the remaining submatrix is empty) or the array ends (the previous player has chosen the last element), then he loses.

Your task is to determine the winner if the players play optimally.

Input

The first line of the input contains t ($1 \le t \le 300$) – the number of test cases.

The first line of each test case contains three integers l, n, and m ($1 \le l, n, m \le 300$) – the size of the array and the sizes of the matrix.

The second line contains l integers $a_1, a_2, a_3, \ldots a_l$ ($1 \le a_i \le \min(7, n \cdot m)$) – the elements of the array a.

The i-th of the last n lines contains m integers $b_{i,1}, b_{i,2}, b_{i,3}, \ldots b_{i,m}$ ($1 \le b_{i,j} \le \min(7, n \cdot m)$) – representing the i-th row of the matrix.

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

It is guaranteed that the sum of l over all test cases does not exceed 300.

Output

You should output t lines, the i-th of them containing a character representing the answer of the i-th test case: "T" if Tsovak wins or "N", otherwise (without quotes).

input

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

output

```
N
T
N
```

In the first example, Tsovak starts by looking for 1. There is only one occurrence of 1 at $(1, 1)$, so he chooses it. Then Narek needs to look for 2 in the submatrix of $(2, 2)$, which consists of just the last two elements: 5 and 2. He chooses 2, and then Tsovak loses since the array has ended.

In the second example, Tsovak needs to choose 1. There is a 1 at the cell (n, m), so he chooses that cell. Then, since the submatrix of $(n + 1, m + 1)$ is empty, Narek cannot find 2, so he loses.

E2. Subtangle Game (Hard Version)

2 seconds, 256 megabytes

This is the hard version of the problem. The differences between the two versions are the constraints on all the variables. You can make hacks only if both versions of the problem are solved.

Tsovak and Narek are playing a game. They have an array a and a matrix b of integers with n rows and m columns, numbered from 1. The cell in the i-th row and the j-th column is (i, j).

They are looking for the elements of a in turns; Tsovak starts first. Each time a player looks for a cell in the matrix containing the current element of a (Tsovak looks for the first, then Narek looks for the second, etc.). Let's say a player has chosen the cell (r, c). The next player has to choose his cell in the submatrix starting at $(r + 1, c + 1)$ and ending in (n, m) (the submatrix can be empty if $r = n$ or $c = m$). If a player cannot find such a cell (or the remaining submatrix is empty) or the array ends (the previous player has chosen the last element), then he loses.

Your task is to determine the winner if the players play optimally.

Note: since the input is large, you may need to optimize input/output for this problem.

For example, in C++, it is enough to use the following lines at the start of the *main()* function:

```
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);
}
```

Input

The first line of the input contains t ($1 \le t \le 1500$) – the number of test cases.

The first line of each test case contains three integers l, n, and m ($1 \le l, n, m \le 1500$) – the size of the array and the sizes of the matrix.

The second line contains l integers $a_1, a_2, a_3, \ldots a_l$ ($1 \le a_i \le n \cdot m$) – the elements of the array a.

The i-th of the last n lines contains m integers $b_{i,1}, b_{i,2}, b_{i,3}, \ldots b_{i,m}$ ($1 \le b_{i,j} \le n \cdot m$) – representing the i-th row of the matrix.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^6$.

It is guaranteed that the sum of l over all test cases does not exceed 1500.

Output

You should output t lines, the i-th of them containing a character representing the answer of the i-th test case: "T" if Tsovak wins or "N", otherwise (without quotes).

input

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

output

```
N
T
N
```

In the first example, Tsovak starts by looking for 1. There is only one occurrence of 1 at $(1,1)$, so he chooses it. Then Narek needs to look for 2 in the submatrix of $(2,2)$, which consists of just the last two elements: 6 and 2. He chooses 2, and then Tsovak loses since the array has ended.

In the second example, Tsovak needs to choose 1. There is a 1 at the cell (n,m), so he chooses that cell. Then, since the submatrix of $(n+1,m+1)$ is empty, Narek cannot find 2, so he loses.