CodeCraft-20 (Div. 2)

A. Grade Allocation

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

n students are taking an exam. The highest possible score at this exam is m. Let a_i be the score of the i-th student. You have access to the school database which stores the results of all students.

You can change each student's score as long as the following conditions are satisfied:

- All scores are integers
- $0 \le a_i \le m$
- The average score of the class doesn't change.

You are student 1 and you would like to maximize your own score.

Find the highest possible score you can assign to yourself such that all conditions are satisfied.

Input

Each test contains multiple test cases.

The first line contains the number of test cases t ($1 \le t \le 200$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 10^3$, $1 \le m \le 10^5$) — the number of students and the highest possible score respectively.

The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le m$) — scores of the students.

Output

For each testcase, output one integer — the highest possible score you can assign to yourself such that both conditions are satisfied._

Example

input
2 4 10 1 2 3 4 4 5 1 2 3 4
output
10 5

Note

In the first case, $a = [1, 2, 3, 4]$, with average of 2.5. You can change array a to $[10, 0, 0, 0]$. Average remains 2.5, and all conditions are satisfied.

In the second case, $0 \le a_i \le 5$. You can change a to $[5, 1, 1, 3]$. You cannot increase a_1 further as it will violate condition $0 \le a_i \le m$.

B. String Modification

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has a string s of length n. He decides to make the following modification to the string:

1. Pick an integer k, ($1 \le k \le n$).
2. For i from 1 to $n-k+1$, reverse the substring $s[i:i+k-1]$ of s. For example, if string s is `qwer` and $k = 2$, below is the series of transformations the string goes through:
    - `qwer` (original string)
    - `wqer` (after reversing the first substring of length 2)
    - `weqr` (after reversing the second substring of length 2)
    - `werq` (after reversing the last substring of length 2)

Hence, the resulting string after modifying s with $k = 2$ is `werq`.

Vasya wants to choose a k such that the string obtained after the above-mentioned modification is lexicographically smallest possible among all choices of k. Among all such k, he wants to choose the smallest one. Since he is busy attending Felicity 2020, he asks for your help.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

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 a single integer n ($1 \leq n \leq 5000$) — the length of the string s.

The second line of each test case contains the string s of n lowercase latin letters.

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

Output
For each testcase output two lines:

In the first line output the lexicographically smallest string s' achievable after the above-mentioned modification.

In the second line output the appropriate value of k ($1 \leq k \leq n$) that you chose for performing the modification. If there are multiple values of k that give the lexicographically smallest string, output the smallest value of k among them.

Example

input
6 4 abab 6 qwerty 5 aaaaa 6 alaska 9 lfpbavjsm 1 p
output
abab 1 ertyqw 3 aaaaa 1 aksala 6 avjsmbpfl 5 p 1

Note
In the first testcase of the first sample, the string modification results for the sample abab are as follows :

- for $k = 1$: abab
- for $k = 2$: baba
- for $k = 3$: abab
- for $k = 4$: baba

The lexicographically smallest string achievable through modification is abab for $k = 1$ and 3. Smallest value of k needed to achieve is hence 1.

C. Primitive Primes

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is Professor R's last class of his teaching career. Every time Professor R taught a class, he gave a special problem for the students to solve. You being his favourite student, put your heart into solving it one last time.

You are given two polynomials $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $g(x) = b_0 + b_1x + \cdots + b_{m-1}x^{m-1}$, with positive integral coefficients. It is guaranteed that the cumulative GCD of the coefficients is equal to 1 for both the given polynomials. In other words, $gcd(a_0, a_1, ..., a_{n-1}) = gcd(b_0, b_1, ..., b_{m-1}) = 1$. Let $h(x) = f(x) \cdot g(x)$. Suppose that $h(x) = c_0 + c_1x + \cdots + c_{n+m-2}x^{n+m-2}$.

You are also given a prime number p. Professor R challenges you to find any t such that c_t isn't divisible by p. He guarantees you that under these conditions such t always exists. If there are several such t, output any of them.

As the input is quite large, please use fast input reading methods.

Input

The first line of the input contains three integers, n, m and p ($1 \le n, m \le 10^6, 2 \le p \le 10^9$), — n and m are the number of terms in $f(x)$ and $g(x)$ respectively (one more than the degrees of the respective polynomials) and p is the given prime number.

It is guaranteed that p is prime.

The second line contains n integers $a_0, a_1, ..., a_{n-1}$ ($1 \le a_i \le 10^9$) — a_i is the coefficient of x^i in $f(x)$.

The third line contains m integers $b_0, b_1, ..., b_{m-1}$ ($1 \le b_i \le 10^9$) — b_i is the coefficient of x^i in $g(x)$.

Output

Print a single integer t ($0 \le t \le n + m - 2$) — the appropriate power of x in $h(x)$ whose coefficient isn't divisible by the given prime p. If there are multiple powers of x that satisfy the condition, print any.

Examples

input
3 2 2 1 1 2 2 1
output
1

input
2 2 999999937 2 1 3 1
output
2

Note

In the first test case, $f(x)$ is $2x^2 + x + 1$ and $g(x)$ is $x + 2$, their product $h(x)$ being $2x^3 + 5x^2 + 3x + 2$, so the answer can be 1 or 2 as both 3 and 5 aren't divisible by 2.

In the second test case, $f(x)$ is $x + 2$ and $g(x)$ is $x + 3$, their product $h(x)$ being $x^2 + 5x + 6$, so the answer can be any of the powers as no coefficient is divisible by the given prime.

D. Nash Matrix

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nash designed an interesting yet simple board game where a player is simply required to follow instructions written on the cell where the player currently stands.

This board game is played on the $n \times n$ board. Rows and columns of this board are numbered from 1 to n. The cell on the intersection of the r-th row and c-th column is denoted by (r, c).

Some cells on the board are called **blocked zones**. On each cell of the board, there is written one of the following 5 characters — U, D, L, R or X — instructions for the player. Suppose that the current cell is (r, c). If the character is R, the player should move to the right cell $(r, c + 1)$, for L the player should move to the left cell $(r, c - 1)$, for U the player should move to the top cell $(r - 1, c)$, for D the player should move to the bottom cell $(r + 1, c)$. Finally, if the character in the cell is X, then this cell is the **blocked zone**. The player should remain in this cell (the game for him isn't very interesting from now on).

It is guaranteed that the characters are written in a way that the player will never have to step outside of the board, no matter at which cell he starts.

As a player starts from a cell, he moves according to the character in the current cell. The player keeps moving until he lands in a blocked zone. It is also possible that the player will keep moving infinitely long.

For every of the n^2 cells of the board Alice, your friend, wants to know, how will the game go, if the player starts in this cell. For each starting cell of the board, she writes down the cell that the player stops at, or that the player never stops at all. She gives you the information she has written: for each cell (r, c) she wrote:

- a pair (x,y), meaning if a player had started at (r, c), he would end up at cell (x,y).

- or a pair $(-1,-1)$, meaning if a player had started at (r, c), he would keep moving infinitely long and would never enter the blocked zone.

It might be possible that Alice is trying to fool you and there's no possible grid that satisfies all the constraints Alice gave you. For the given information Alice provided you, you are required to decipher a possible board, or to determine that such a board doesn't exist. If there exist several different boards that satisfy the provided information, you can find any of them.

Input

The first line of the input contains a single integer n $(1 \le n \le 10^3)$ — the side of the board.

The i-th of the next n lines of the input contains $2n$ integers $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$, where (x_j, y_j) $(1 \le x_j \le n, 1 \le y_j \le n$, or $(x_j, y_j) = (-1, -1))$ is the pair written by Alice for the cell (i, j).

Output

If there doesn't exist a board satisfying the information that Alice gave you, print a single line containing `INVALID`.

Otherwise, in the first line print `VALID`. In the i-th of the next n lines, print the string of n characters, corresponding to the characters in the i-th row of the suitable board you found. Each character of a string can either be U, D, L, R or X. If there exist several different boards that satisfy the provided information, you can find any of them.

Examples

input
2 1 1 1 1 2 2 2 2
output
VALID XL RX

input
3 -1 -1 -1 -1 -1 -1 -1 -1 2 2 -1 -1 -1 -1 -1 -1 -1 -1
output
VALID RRD UXD ULL

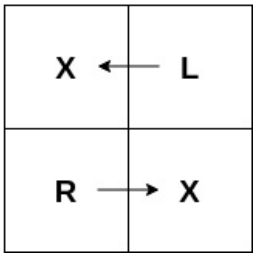
Note

For the sample test 1 :

The given grid in output is a valid one.

- If the player starts at $(1, 1)$, he doesn't move any further following X and stops there.
- If the player starts at $(1, 2)$, he moves to left following L and stops at $(1, 1)$.
- If the player starts at $(2, 1)$, he moves to right following R and stops at $(2, 2)$.
- If the player starts at $(2, 2)$, he doesn't move any further following X and stops there.

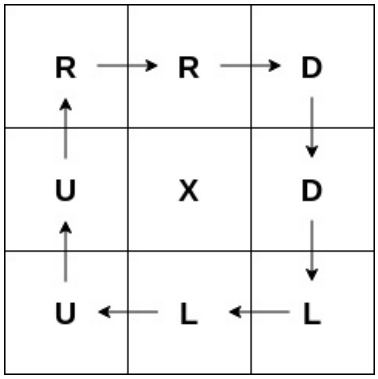
The simulation can be seen below :



For the sample test 2 :

The given grid in output is a valid one, as a player starting at any cell other than the one at center $(2, 2)$, keeps moving in an infinitely long cycle and never stops. Had he started at $(2, 2)$, he wouldn't have moved further following instruction X .

The simulation can be seen below :



E. Team Building

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice, the president of club FCB, wants to build a team for the new volleyball tournament. The team should consist of p players playing in p different positions. She also recognizes the importance of audience support, so she wants to select k people as part of the audience.

There are n people in Byteland. Alice needs to select exactly p players, one for each position, and exactly k members of the audience from this pool of n people. Her ultimate goal is to maximize the total strength of the club.

The i-th of the n persons has an integer a_i associated with him — the strength he adds to the club if he is selected as a member of the audience.

For each person i and for each position j, Alice knows $s_{i,j}$ — the strength added by the i-th person to the club if he is selected to play in the j-th position.

Each person can be selected at most once as a player or a member of the audience. You have to choose exactly one player for each position.

Since Alice is busy, she needs you to help her find the maximum possible strength of the club that can be achieved by an optimal choice of players and the audience.

Input

The first line contains 3 integers n, p, k ($2 \le n \le 10^5$, $1 \le p \le 7$, $1 \le k, p + k \le n$).

The second line contains n integers $a_1, a_2, \ldots, a_n$. ($1 \le a_i \le 10^9$).

The i-th of the next n lines contains p integers $s_{i,1}, s_{i,2}, \ldots, s_{i,p}$. ($1 \le s_{i,j} \le 10^9$)

Output

Print a single integer res — the maximum possible strength of the club.

Examples

input
4 1 2 1 16 10 3 18 19 13 15
output
44

input
6 2 3 78 93 9 17 13 78 80 97 30 52 26 17 56 68 60 36 84 55
output
377

input
3 2 1 500 498 564

100002 3
422332 2
232323 1

output

422899

Note

In the first sample, we can select person 1 to play in the 1-st position and persons 2 and 3 as audience members. Then the total strength of the club will be equal to $a_2 + a_3 + s_{1,1}$.

F. Battalion Strength

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n officers in the Army of Byteland. Each officer has some power associated with him. The power of the i-th officer is denoted by p_i. As the war is fast approaching, the General would like to know the strength of the army.

The strength of an army is calculated in a strange way in Byteland. The General selects a random subset of officers from these n officers and calls this subset a battalion.(All 2^n subsets of the n officers can be chosen equally likely, including empty subset and the subset of all officers).

The strength of a battalion is calculated in the following way:

Let the powers of the chosen officers be $a_1, a_2, \ldots, a_k$, where $a_1 \le a_2 \le \cdots \le a_k$. The strength of this battalion is equal to $a_1a_2 + a_2a_3 + \cdots + a_{k-1}a_k$. (If the size of Battalion is ≤ 1, then the strength of this battalion is 0).

The strength of the army is equal to the expected value of the strength of the battalion.

As the war is really long, the powers of officers may change. Precisely, there will be q changes. Each one of the form $i\ x$ indicating that p_i is changed to x.

You need to find the strength of the army initially and after each of these q updates.

Note that the changes are permanent.

The strength should be found by modulo $10^9 + 7$. Formally, let $M = 10^9 + 7$. It can be shown that the answer can be expressed as an irreducible fraction p/q, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Input

The first line of the input contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of officers in Byteland's Army.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$).

The third line contains a single integer q ($1 \le q \le 3 \cdot 10^5$) — the number of updates.

Each of the next q lines contains two integers i and x ($1 \le i \le n$, $1 \le x \le 10^9$), indicating that p_i is updated to x.

Output

In the first line output the initial strength of the army.

In i-th of the next q lines, output the strength of the army after i-th update.

Examples

input

2
1 2
2
1 2
2 1

output

500000004
1
500000004

input

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

output

```
625000011
13
62500020
375000027
62500027
```

Note

In first testcase, initially, there are four possible battalions

- $\{\}$ Strength = 0
- $\{1\}$ Strength = 0
- $\{2\}$ Strength = 0
- $\{1, 2\}$ Strength = 2

So strength of army is $\frac{0+0+0+2}{4} = \frac{1}{2}$
After changing p_1 to 2, strength of battallion $\{1, 2\}$ changes to 4, so strength of army becomes 1.

After changing p_2 to 1, strength of battalion $\{1, 2\}$ again becomes 2, so strength of army becomes $\frac{1}{2}$.