Codeforces Round #416 (Div. 2)

A. Vladik and Courtesy

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

At regular competition Vladik and Valera won a and b candies respectively. Vladik offered 1 his candy to Valera. After that Valera gave Vladik 2 his candies, so that no one thought that he was less generous. Vladik for same reason gave 3 candies to Valera in next turn.

More formally, the guys take turns giving each other one candy more than they received in the previous turn.

This continued until the moment when one of them couldn't give the right amount of candy. Candies, which guys got from each other, they **don't consider as their own**. You need to know, who is the first who can't give the right amount of candy.

Input

Single line of input data contains two space-separated integers a, b $(1 \le a, b \le 10^9)$ — number of Vladik and Valera candies respectively.

Output

Pring a single line "`Vladik`" in case, if Vladik first who can't give right amount of candy, or "`Valera`" otherwise.

Examples

input
1 1
output
Valera

input
7 6
output
Vladik

Note

Illustration for first test case:

step	Vladik	Valera
0	1	1
1	0	1

Illustration for second test case:

step	Vladik	Valera
0	7	6
1	6	6
2	6	4
3	3	4
4	3	0

B. Vladik and Complicated Book

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

Vladik had started reading a complicated book about algorithms containing n pages. To improve understanding of what is written, his friends advised him to read pages in some order given by permutation $P = [p_1, p_2, \ldots, p_n]$, where p_i denotes the number of page that should be read i-th in turn.

Sometimes Vladik's mom sorted some subsegment of permutation P from position l to position r inclusive, because she loves the order. For every of such sorting Vladik knows number x — what index of page in permutation he should read. He is wondered if the page, which he will read after sorting, has changed. In other words, has p_x changed? After every sorting Vladik return permutation to initial state, so you can assume that each sorting is independent from each other.

Input

First line contains two space-separated integers n, m $(1 \le n, m \le 10^4)$ — length of permutation and number of times Vladik's mom sorted some subsegment of the book.

Second line contains n space-separated integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$ — permutation P. Note that elements in permutation are distinct.

Each of the next m lines contains three space-separated integers l_i, r_i, x_i $(1 \le l_i \le x_i \le r_i \le n)$ — left and right borders of sorted subsegment in i-th sorting and position that is interesting to Vladik.

Output

For each mom's sorting on it's own line print "Yes", if page which is interesting to Vladik hasn't changed, or "No" otherwise.

Examples

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

output
Yes No Yes Yes No

input
6 5 1 4 3 2 5 6 2 4 3 1 6 2 4 5 4 1 3 3 2 6 3

output
Yes No Yes No Yes

Note

Explanation of first test case:

1. $[1, 2, 3, 4, 5]$ — permutation after sorting, 3-rd element hasn't changed, so answer is "Yes".
2. $[3, 4, 5, 2, 1]$ — permutation after sorting, 1-st element has changed, so answer is "No".
3. $[5, 2, 3, 4, 1]$ — permutation after sorting, 3-rd element hasn't changed, so answer is "Yes".
4. $[5, 4, 3, 2, 1]$ — permutation after sorting, 4-th element hasn't changed, so answer is "Yes".
5. $[5, 1, 2, 3, 4]$ — permutation after sorting, 3-rd element has changed, so answer is "No".

C. Vladik and Memorable Trip

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

Vladik often travels by trains. He remembered some of his trips especially well and I would like to tell you about one of these trips:

Vladik is at initial train station, and now n people (including Vladik) want to get on the train. They are already lined up in some order, and for each of them the city code a_i is known (the code of the city in which they are going to).

Train chief selects some number of disjoint segments of the original sequence of people (covering entire sequence by segments is **not necessary**). People who are in the same segment will be in the same train carriage. The segments are selected in such way that if at least one person travels to the city x, then all people who are going to city x should be in the same railway carriage. This means that they can't belong to different segments. Note, that all people who travel to the city x, either go to it and in the same railway carriage, or do not go anywhere at all.

Comfort of a train trip with people on segment from position l to position r is equal to XOR of all distinct codes of cities for people on the segment from position l to position r. XOR operation also known as exclusive OR.

Total comfort of a train trip is equal to sum of comfort for each segment.

Help Vladik to know maximal possible total comfort.

Input

First line contains single integer n ($1 \le n \le 5000$) — number of people.

Second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 5000$), where a_i denotes code of the city to which i-th person is going.

Output

The output should contain a single integer — maximal possible total comfort.

Examples

input
6 4 4 2 5 2 3
output
14

input
9 5 1 3 1 5 2 4 2 5
output
9

Note

In the first test case best partition into segments is: [4, 4] [2, 5, 2] [3], answer is calculated as follows: $4 + (2\ xor\ 5) + 3 = 4 + 7 + 3 = 14$

In the second test case best partition into segments is: 5 1 [3] 1 5 [2, 4, 2] 5, answer calculated as follows: $3 + (2\ xor\ 4) = 3 + 6 = 9$.

D. Vladik and Favorite Game

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

This is an interactive problem.

Vladik has favorite game, in which he plays all his free time.

Game field could be represented as $n \times m$ matrix which consists of cells of three types:

- «`.`» — normal cell, player can visit it.
- «`F`» — finish cell, player has to finish his way there to win. There is exactly one cell of this type.
- «`*`» — dangerous cell, if player comes to this cell, he loses.

Initially player is located in the left top cell with coordinates $(1, 1)$.

Player has access to 4 buttons "`U`", "`D`", "`L`", "`R`", each of them move player up, down, left and right directions respectively.

But it's not that easy! Sometimes friends play game and change functions of buttons. Function of buttons "`L`" and "`R`" could have been swapped, also functions of buttons "`U`" and "`D`" could have been swapped. Note that functions of buttons can be changed only at the beginning of the game.

Help Vladik win the game!

Input

First line contains two space-separated integers n and m $(1 \le n, m \le 100)$ — number of rows and columns respectively.

Each of next n lines contains m characters describing corresponding row of field. Set of characters in field is described above.

Guaranteed that cell with coordinates $(1, 1)$ is normal and there is at least one way from initial cell to finish cell without dangerous cells.

Interaction

You can press buttons no more than $2 \cdot n \cdot m$ times.

To press a button you should print "`U`", "`D`", "`L`", "`R`" in new line. It's necessary to print newline character and `flush` output. After flushing buffer you should read answer from input data. Answer is the pair of space-separated integers x, y — new position of player. In case, if there is no cell in direction of moving, position will not change. If after any move player lost, in other words player move to dangerous cell, then x and y will be equal to -1.

If after any move player is in finish or dangerous cell, then you should terminate your program.

To finish output buffer (i. e. for operation `flush`) right after printing direction and newline you should do next:

- `fflush(stdout)` in C++
- `System.out.flush()` in Java
- `stdout.flush()` in Python
- `flush(output)` in Pascal
- read documentation for other languages.

Hacks

To perform a hack you should use this format:

```
n m swapLR swapUD
a_1
a_2
...
a_n
```

Where n, m — number of rows and columns in game field. $swapLR$ is equal to 1 in case, when directions "`L`" and "`R`" is swapped, and equal to 0 otherwise. $swapUD$ is equal to 1, when directions "`U`" and "`D`" is swapped, and equal to 0 otherwise. a_1, a_2, ..., a_n — description of corresponding rows of game field.

Example

input
4 3 ... **. F*. ... 1 1 1 2 1 3 1 3

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

output

```
R
L
L
D
U
U
U
R
R
D
```

Note

In first test case all four directions swapped with their opposite directions. Protocol of interaction In more convenient form:

Judge	Contestant
4 3 ... **. F*. ...	
	R
1 1	
	L
1 2	
	L
1 3	
	D
1 3	
	U
2 3	
	U
3 3	
	U
4 3	
	R
4 2	
	R
4 1	
	D
3 1	

This test could be presenter for hack in following way:

```
4 3 1 1
...
**.
F*.
...
```

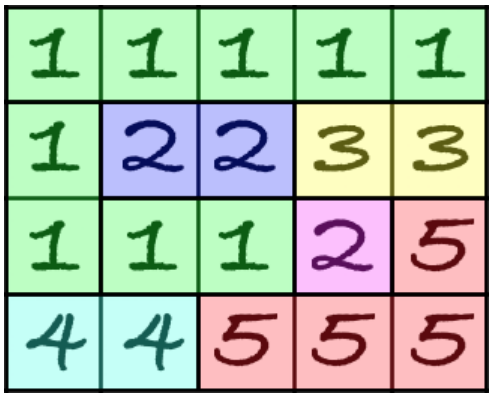
E. Vladik and Entertaining Flags

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

In his spare time Vladik estimates beauty of the flags.

Every flag could be represented as the matrix $n \times m$ which consists of positive integers.

Let's define the beauty of the flag as number of components in its matrix. We call component a set of cells with same numbers and between any pair of cells from that set there exists a path through adjacent cells from same component. Here is the example of the partitioning some flag matrix into components:



But this time he decided to change something in the process. Now he wants to estimate not the entire flag, but some segment. Segment of flag can be described as a submatrix of the flag matrix with opposite corners at $(1, l)$ and (n, r), where conditions $1 \le l \le r \le m$ are satisfied.

Help Vladik to calculate the beauty for some segments of the given flag.

Input

First line contains three space-separated integers n, m, q $(1 \le n \le 10, 1 \le m, q \le 10^5)$ — dimensions of flag matrix and number of segments respectively.

Each of next n lines contains m space-separated integers — description of flag matrix. All elements of flag matrix is positive integers not exceeding 10^6.

Each of next q lines contains two space-separated integers l, r $(1 \le l \le r \le m)$ — borders of segment which beauty Vladik wants to know.

Output

For each segment print the result on the corresponding line.

Example

input
4 5 4 1 1 1 1 1 1 2 2 3 3 1 1 1 2 5 4 4 5 5 5 1 5 2 5 1 2 4 5
output
6 7 3 4

Note

Partitioning on components for every segment from first test case: