Task A. Pyramid

Input file name:	`Standard input`
Output file name:	`Standard output`
Time limit:	1 second
Memory limit:	64 MB

A 2D pyramid is built out of rectangular blocks. Each block has a width and a height. You can place one block upon another only if the width of the upper block is strictly less than the width of the lower one. The lowest block of the pyramid can have any width. Determine the biggest possible height of a pyramid based on a set number of blocks.

Input format

The first line of input data contains N which is the number of blocks (**1** 6 *N* 6 **100,000)**. In the next N of lines, width W_i and height h_i of a block are set (**1** 6 W_i, h_i 6 $\mathbf{10}^9$).

Output format

The maximum height of the pyramid should be the output. It should be one whole number.

Examples

Standard input	Standard output

`3`
`3 1`
`2 2`
`33`
	`5`

Note

In this example, a pyramid will consist of two blocks. Block 3 will be the lower one, block 2 the upper one. Blocks 1 and 3 cannot be used together.

Evaluation criteria

Solutions working appropriately at **1** 6 *N*, W_i, h_i 6 **100** will get at least 50% of points.

Task B. Bonusman

Input file name:	`Standard input`
Output file name:	`Standard output`
Time limit:	15 seconds
Memory limit:	512 MB

A playing field in the Bonusman game consists of *N* rows и *M* columns. Each block of the playing field can either be a wall (W), a bonus (B), or empty (.)
A player can place a bonus trap in an empty block. The bonus trap collects all bonuses located in the same row or column until there is a wall or an edge of the playing field in the way. It means that all bonuses accessible moving to the left, right, up, or down are collected unless there is a wall or an edge of the playing field in the way.
Blocks are numbered starting from the upper left hand corner. The starting number is 1.
Calculate in which block the player should place a bonus trap to collect as many bonuses as possible.

Input format

Numbers N (1 6 N 6 1,000) and M (1 6 M 6 1,000) are set in the first line.
The next N lines contain the description of the playing field. Each line consists of M symbols (B), (W), or (.)

Output format

Output the row and number column where, if placed, the bonus trap will collect the biggest amount of bonuses. If there are several answers possible, output any of them.

Examples

Standard input	Standard output

`3 3`
`WBW`
`B. B`
`WBW`
	`2 2`

`3 4`	`11`

`1 10`
`. BBWBB. BBW`
	`1 7`

Note

Test 1:3 3 WBW B. B. WBW Answer 1:2 2 Test 2:3 4 Answer 2: 1 1 Test 3: 1 10. BBWBB. BBW Answer 3: 1 7 Note: in test 2, the bonus trap can be located anywhere. The number of bonuses collected will still be 0.

Task C. Filler

Input file name:	`Standard input`
Output file name:	`Standard output`
Time limit:	2 seconds
Memory limit:	64 MB

The Filler game takes place at a rectangular playing field. Each block is coloured in one of six colours. The game starts in the block located in the upper left-hand corner. During each turn, the player can change the colour of the area containing the upper left block. As a result of that, the adjacent areas of the same colour will connect with it. Areas that have two blocks adjacent to each other are considered adjacent. An area is a set of blocks of the same color where you can get from one block into another by moving across the side of the block.
The objective is to occupy the lower right block by having the least amount of recolorings.
Determine the amount and order of recolorings.

Input format

Numbers N (1 6 N 6 2,000) and M (1 6 M 6 2,000) are put in. These are the dimensions of the playing field.
Each of the next N lines contains M numbers from 1 to 6. They represent the colours of the blocks.

Output format

Output K in the first line. This is the least amount of recolorings needed to reach the lower right block.
The second line should contain K numbers from 1 to 6. This is the sequence of recolorings.
If there are several possible sequences of recolorings, output any of them.

Example

Standard input	Standard output

3 5
1 2 3 5 6
3 4 3 6 6
4 3 16 5
	4

Note

Example 1:35123563436643165 Answer 1: 4 2 3 6 5

Task D. Distribution according to majors

Input file name:	`Standard input`
Output file name:	`Standard output`
Time limit:	4 seconds
Memory limit:	256 MB

Having completed the second year of education at HSE's Faculty of Computer Science, faculty students select their majors. They make a list of priority majors they would like to select. Each student is ranked based on their GPAs throughout the duration of studies. The ranking of some students may coincide. There is a specific maximum number of students that may be admitted to pursue a specific major.
Help to distribute students according to their wishes. The following condition should be met for every student. After a student is assigned to a major of higher priority for them, there should not be a single student ranked higher than them. Out of all options, we're interested in the one where the most number of students are assigned a major.

Input format

The first line contains two numbers, N and K (**1** 6 N, K 6 **100000**). These are the number of students and majors respectively.
The next line contains K of natural numbers Si (**1** 6 S_i 6 **100000**) where S_i is the number of students that can be assigned a major i.
The next N lines contain the description of priorities for students. It consists of R (**1** 6 R 6 N), or student's ranking, T (**1** 6 T 6 N), or the amount of desirable majors, and T numbers between 1 and K, or the number of desirable majors in descending order of priority. Major numbers are unique. It is guaranteed that the sum of T for all students does not exceed $\mathbf{10}^6$.

Output format

For each of N students, output the number of the major they will be assigned. In case all spots at prioritized majors are occupied by students with a higher ranking, number 1 should be output for that student. If there are several right answers, output any one of them.

Example

Standard input	Standard output

`4 2`
`1 5`
`3 11`
`112`
`2212`
`3212`
	`-12 12`

Note

Solutions that work correctly if the number of students and majors does not exceed 100 will receive at least 50% of the points.

Task E. Map description

Input file name:	`Standard input`
Output file name:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 MB

In ancient times, there used to be very few maps. In order to find out how to get from one town to another, one would rely on the description provided by fellow travellers.
Only *N* number of towns were known to exist. They were numbered from 1 to *N*. At first, nothing was known about whether these towns were connected with roads. Then, travellers would tell whether it was possible or not to get from one town to another. Some also requested the way to get around. All roads are two-sided, that is, if it is possible/impossible to get from one town to another, the reverse is also true. Help scientists researching the ancient road networks answer these requests. The requests should be processed chronologically using only the information available at the time of the request. The road network did not change during the research. Travellers' reports do not contradict each other.

Input format

The first input line contains two numbers, N and *K* (**1** 6 *N*, *K* 6 **100000**). These are the number of towns and requests.
Each of the following *K* lines should contain a request of one of three types:
+ i j – there is a road between towns i and j.
- i j – there is no road between towns and j.
? i j – determine whether there is a road between towns i and j.

Output format

Output one of these three lines as a reply to each request:
+ – if a road exist
- – if a road does not exist
? – if, at the time of the request, there is no certain way of knowing whether a road exists

Examples

Standard input	Standard output

`3 4`
`+ 12`
`? 1 3`
`+32`
`? 1 3`
	`?`
	`+`

`4 5`
`+12`
`+34`
`- 1 4`
`? 2 4`
`? 1 3`
	`-`
	`-`

Note

Solutions working appropriately at **1** 6 *N*, *K* 6 **1000** will get at least 50% of points.