Problem A

Time travel

Suppose you had a time machine that could be used at most three times. Each time you could choose to go back to the past or go to the future. The machine has three fixed credits, which are a certain amount of years. You can travel once, twice or three times, and each credit can be used only once. For example, if the credits were $5, 12$ and 9, you could decide to travel twice: go 5 years to the future and then come back 9 years to the past. This way, you would end up four years in the past, in 2012. You could also travel three times, all to the future, using the credits in any order, ending up in 2042.

In this problem, given the values of the three credits, your program must decide if it is possible, or not, to travel in time and come back to the present, using at least one credit and, at most, three credits; always using each one of the three credits only once.

Input

The only line of the input contains three integers A, B and C ($1 \leq A, B, C \leq 1000$), representing the credit values.

Output

Output one line containing the character "`S`" if it is possible to travel in time and come back to the present, or "`N`" otherwise.

Examples

Examples

Input	Output

`22 5 22`
`31 110 79`
`45 8 7`
	`S`
	`S`
	`N`

Problem B

Hot Potato

Hot potato is a popular game among children in school. The game is simple: on each turn, the child who has the potato passes it to another child. In some point, the teacher, which is not looking at the game will go and say the game is over. When this happens, the child who has the potato loses the game.

A variation of this game, played in the cafeteria is proposed by the teacher. Children are numbered from 1 to N according to their position in the queue, where the child with the number 1 is the first on the queue. Each child receives a paper with a number and during the game whenever a child receives the potato the child will pass it to the child in the position noted in his paper. If the potato gets to a position less or equals to X in the queue, where X is defined at the beginning of the game, the game ends and the teacher wins the game. If this never happens, the game never ends and the children will win the game: The next day all they get a discount in the cafeteria.

The teacher starts the game passing the potato to some child in the queue. As his sight is not very good, he can only guarantee that he will pass the potato to some child in a range $L \dots R$ in the queue with the same probability. He is considering different possible ranges to start the game. For this, the teacher would like to know, for each of these ranges, which X value he must choose so that the game can be the most fair as possible, this is, the probability that the game ends is as close as possible to the probability of the game not finishing.

You should help the teacher to evaluate the proposals. Given the papers of each child in the queue and several possible ranges, you should answer, for each range, the value of X that makes the fairest game possible. If there is a tie, answer the X closer to the front of the queue.

Input

The input contains several test cases. In each test case, The first line of input contains two integer numbers, N and Q ($2 \leq N \leq 50000$, $1 \leq Q \leq 10^5$). The next line contains N integers $p_1, p_2 \dots p_N$ ($1 \leq p_i \leq N$), the numbers on the papers each child receives. The next Q lines, each contain two integers L and R ($1 \leq L \leq R \leq N$), the range that is being considered by the teacher.

Output

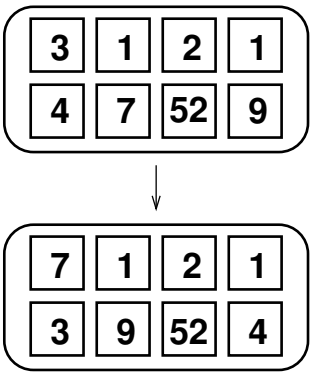
For each test case in the input, You should print Q lines, each line contains, for each range considered by the teacher, the number X that must be choosen by the teacher so that the game is the fairest as possible.

Examples

Input	Output
9 4 2 3 4 5 6 7 4 9 5 1 3 3 5 2 8 7 9 3 3 1 3 3 1 1 1 2 2 3	1 3 3 1 1 1 2

Problem C

Containers



The CBS–Container Balancing System needs to be updated so as to work with a new class of ships, the "two by four", which are ships that can carry eight large containers disposed in two lines and four columns, exactly as shown in the figure. These ships have a fixed crane that can perform a single type of movement: picking up two adjacent containers, in a row or column, and exchange their position. To speed up the loading process in the harbor, the eight containers are placed in any of the eight positions, defining an initial configuration. When ship leaves the harbor, the crane needs to move the containers so they end in a predefined final configuration.

The problem is that the cost for the crane to perform one movement is equal to the sum of the weights of the two adjacent containers whose positions are being exchanged. Given the weights of the containers in each position of both the initial and final configurations, the CBS has to compute the minimum possible total cost for a sequence of movements that leads from the initial to the final configuration.

Input

The input consists of four lines containing, each one, four integers between 1 and 1000, inclusive. The first two lines define the weights of the initial configuration and the last two lines, the weigths in the final configuration. There is always a solution, as the containers in the initial and final configurations are the same, possibly in different positions.

Output

Output a line containing an integer, representing the minimum possible total cost for a sequence of movements that leads from the initial to the final configuration.

Examples

Examples

Input	Output
3 1 2 1 4 7 52 9 7 1 2 1 3 9 52 4	81 50
1 2 3 4 5 10 7 8 1 2 3 4 5 8 7 10	0
34 5 6 998 4 17 77 84 34 5 6 998 4 17 77 84	

Problem D

Divisors

Think on a positive number n. Now tell me a divisor A of n. Now give me another number B that is not a divisor of n. Now a multiple C. And a non-multiple D. The number that you thought is...

It looks like magic, but it is math!! Is it that, given A, B, C and D, you can find what the original number n is? Note that it may exist more than one solution.

In this problem, given the values of A, B, C, and D you should write a program that determines which is the lowest number n that may have been thought or conclude that there is not a possible value of n.

Input

The input contains several test cases. In each test case, 4 integer numbers A, B, C and D as described above ($1 \leq A, B, C, D \leq 10^9$).

Output

For each test case in the input, In the case there exist at least one number n which A, B, C and D make sense, the program should print a single line with the lowest possible number n, otherwise print -1.

Examples

Input	Output
2 12 8 2 3 4 60 105	4 6

Problem E

Hexadecimal Statistic

Given a sequence of positive integers in hexadecimal representation, for example, $S = [\texttt{9af47c0b}, \texttt{2545557}, \texttt{ff6447979}]$, we define $\texttt{sum}(S)$ as the sum of all elements in S. Now, consider a certain permutation of the 16 hexadecimal digits, for example, $p = [\texttt{4}, \texttt{9}, \texttt{5}, \texttt{a}, \texttt{0}, \texttt{c}, \texttt{f}, \texttt{3}, \texttt{d}, \texttt{7}, \texttt{8}, \texttt{b}, \texttt{1}, \texttt{2}, \texttt{6}, \texttt{e}]$. Beginning with the base sequence S, we can define a transformed sequence $S^{[4]}$, obtained with the removal of all occurences of the hexadecimal digit `4` from all integers in S, $S^{[4]} = [\texttt{9af7c0b}, \texttt{255557}, \texttt{ff67979}]$. Next, we can remove the digit `9` and obtain $S^{[4,9]} = [\texttt{af7c0b}, \texttt{255557}, \texttt{ff677}]$. Following the digit order in the permutation p, we can define in this way, 16 sequences: $S^{[4]}, S^{[4,9]}, S^{[4,9,5]}, \ldots, S^{[4,9,5,a,0,c,f,3,d,7,8,b,1,2,6,e]}$. We are interested in the sum of all elements from these 16 sequences:

$$\texttt{total}(S, p) = \texttt{sum}(S^{[4]}) + \texttt{sum}(S^{[4,9]}) + \texttt{sum}(S^{[4,9,5]}) + \cdots + \texttt{sum}(S^{[4,9,5,a,0,c,f,3,d,7,8,b,1,2,6,e]})$$

Clearly, this total depends on the permutation p used in the sucessive removal. Given a sequence of N positive integers in hexadecimal, you have to compute, considering all possible permutations of the 16 hexadecimal digits: the minimum total, the maximum total and the sum of all totals from all permutations. For the sum of all totals, print the result modulo `3b9aca07` ($10^9 + 7$ on base 10).

Input

The first line of the input contains an integer N, $\texttt{1} \leq N \leq \texttt{3f}$, representing the size of the sequence. The next N lines contain, each one, a positive integer P, $\texttt{0} \leq P \leq \texttt{fffffffff}$, defining the initial sequence S of integers. All numbers in the input are in hexadecimal, with lowercase letters.

Output

Output one line containing three positive integers, in hexadecimal with lowercase letters, representing the minimum total, the maximum total and the sum of all totals considering all possible permutations of the 16 hexadecimal digits.

Examples

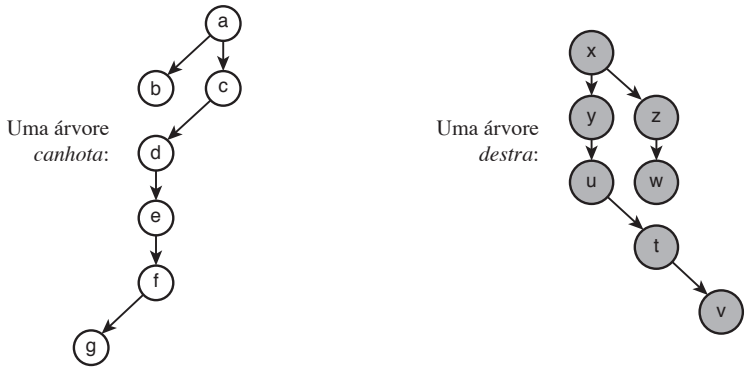
Examples

Input	Output
3 9af47c0b 2545557 ff6447979 1 fffffffff	1312c99c b4e87e9387 5bb5fc 0 effffffff1 15dac189

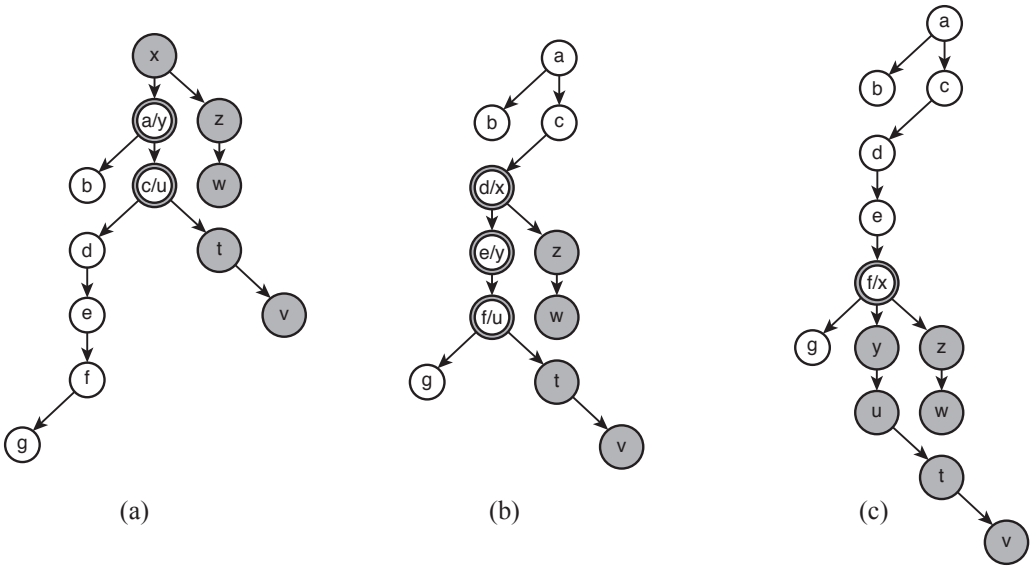
Problem F
Fusing trees

In computer theory, trees are strange objects: the root is at the top and leaves are on the bottom! A tree is a data structure composed of N vertices connected by $N-1$ edtes so that it is possible to get from one vertex to any other vertex following the edges. In a *rooted* tree, every edge connects a *parent* vertex and a *child* vertex. An unique vertex with no parent is named *root*. Thus, from the root it is possible to reach any other tree vertex following the edges in the direction of parent to child.

In a *ternary* tree each vertex can have up to tree child vertex, names *left*, *central*, and *right*. A *left-handed* ternary tree is a rooted ternary tree in which none vertex has a right child. A *right-handed* ternary tree is a rooted ternary tree in which none vertex has a left child. The root of a ternary tree is always a *central* vertex. The following figure shows examples of a left ternary tree and a right ternary tree.



An *overlap* S of a left-handed tree C and a right-handed tree D is a rooted ternary tree where the root is either the root of C, the root of D, or both roots of C and D overlaped, and it contains the structure of both trees overlaped. The figure below shows some trees formed by the overlap of the left-handed and the right-handed trees from the figure above.



Note that on Figure (a) the root is a vertex x (of right-handed tree) and the pairs of vertices (a, y) and (c, u) are overlaped. On figure (b) the root is the vertex a (of left-handed tree) and the pairs of vertices (d, x), (e, y) and (f, u) are overlaped. On figure (c) the root is also the vertex a (of left-handed tree) and the pair of vertices (f, x) are overlaped.

Given a left-handed tree and a right-handed tree, your tasks is to determine the minimum number required of vertices to build a ternary tree that is an overlap of the given trees.

Input

The first line of a test case contains an integer N represengint the number of vertices for the left-handed tree ($1 \leq N \leq 10^4$). Vetices on this tree are identified with the numbers from 1 to N, and the root is the vertex with number 1. Each of the next N lines contains three integer numbers I, L, and K, representing the identifier of vertex I, the identifier for the left child L of I and the identifier for the central child K of I ($0 \leq I, L, K \leq N$). The next line contains an integer M representing the number of vertices of the right-handed tree ($1 \leq M \leq 10^4$). Vertices on this tree are identified with the numbers from 1 to M, and the root is the vertex with number 1. Each of the next M lines contains three integer numbers P, Q, and R, representing the identifier for vertex P, the identifier for the central child Q of P and the identier for the right child R of P. ($0 \leq P, Q, R \leq N$). A value of zero represent an unexistant vertex (used when a vertex does not have one of its children).

Output

Print the minimum number of vertices of a tree that is an overlap of the two trees given in the input.

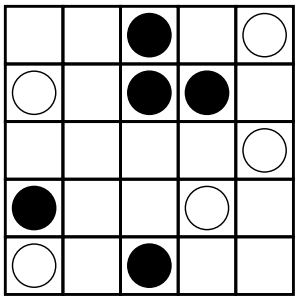
Examples

Input	Output
7 1 2 3 2 0 0 3 4 0 4 0 5 5 0 6 6 7 0 7 0 0 7 1 2 3 2 4 0 3 5 0 4 0 6 5 0 0 6 0 7 7 0 0 5 1 2 3 2 4 5 3 0 0 4 0 0 5 0 0 3 1 2 3 2 0 0 3 0 0 3 3 0 2 2 0 0 1 0 3 2 2 0 0 1 2 0	11 6 3

Problem G

Go−−

Go−− is similar do the traditional game of Go, but it is much simpler! It is played on a square grid of dimension N, initially empty, where two players, one playing with the black stones and the other with the white ones, alternate placing one stone at a time on any of the unoccupied cells. The match ends after each player has placed P stones on the grid. Consider all possible square subgrids of dimensions 1 to N. A square subgrid belongs to the player with the black stones if it contains at least one black stone and no white stones. A square subgrid belongs to the player with the white stones if it contains at least one white stone and no black stone. Note that some subgrids do not belong to any player. Those having no stone at all, or stones of both color.



In this problem, given the final position of the grid, your program has to compute how many square subgrids belong to each player, so we can know who won the match. In the picture, the player who plays with the black stones has 12 subgrids (five of dimension 1, six of dimension 2 and one of dimension 3). The player with the whites stones, who lost the match, has 10 subgrids.

Input

The first line of the input contains two integers N and P, $2 \leq N \leq 500$, $1 \leq P \leq 500$ and $P \leq N^2/2$, representing, respectively, the dimension of the grid and the number of stones each player has placed. Each one of the next P lines contains two integers X and Y defining the coordinates of the black stones. Then, each one of the following P lines contains two integers X and Y defining the coordinates of the white stones. All stones are placed at distinct cells.

Output

Output one line containing two integers indicating how many square subgrids belong to the player with black stones and how many to the player with the white stones.

Examples

Examples

Input	Output
2 1 1 1 2 2 5 5 1 3 2 3 2 4 4 1 5 3 1 5 2 1 3 5 4 4 5 1 500 3 500 498 500 499 500 500 120 124 251 269 499 498	1 1 12 10 4 12463784

Problem H
huaauhahhuahau

It is common among young people to use strings of letters in chats which often seem random to represent laughter. Some common examples are:

```
huaauhahhuahau
hehehehe
ahahahaha
jaisjjkasjksjjskjakijs
huehuehue
```

Claudia is a young programmer who was intrigued by the sound of "digital laughter". She can not even pronounce some of them!! But she realized that some of them seem to show better the feeling of laugther than others. The first thing she noticed is that the consonants do not interfere in how digital laughter influence the transmission of the feeling. The second thing she noticed is that the funniest digital laughs are those in which the vowels are the same when read in its natural order (left to right) and in reverse order (right to left), ignoring the consonants. For example, "hahaha" and "huaauhahhuahau" are some of the funniest laughs, while "riajkdhhihhjak" and "huehuehue" are not.

Claudia is very busy with the statiscal analysis of digital laughter and asked for help to write a program that says, for a digital laugh, if it is one of the funniest or not.

Input

The input contais a sequence with no more than 50 characters, containing only lowercase letters and at least one vowel. Vowels are the letters '`a`','`e`','`i`','`o`','`u`'.

Output

Your program should print "`S`", in case the laugh is one of the funniest, or "`N`" otherwise.

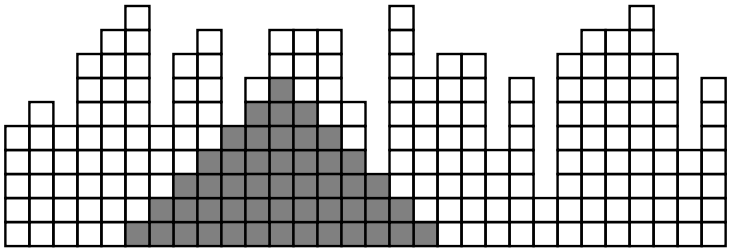
Examples

Input	Output
`hahaha`	`S`
`riajkjdhhihhjak`	`N`
`a`	`S`
`huaauhahhuahau`	`S`

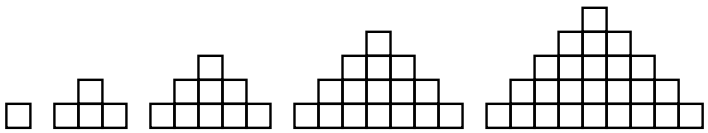
Problem I

Isosceles

Two brothers were playing with wooden blocks trying to build a wall which was still incomplete, with the columns having different heights, as in this picture.



They then decided to remove blocks from the wall, always from the top of the columns, so as to leave a triangle in the end. They can only remove blocks from the wall, not move them from one column to another, and the triangle must be complete. The picture below illustrates the first five triangles, of the type they are interested in, with heights $1, 2, 3, 4$ and 5 respectively.



Given the sequence of column heights in the wall, your program should help the brothers find the maximum possible height for a triangle in the end. For the wall shown in the first figure, with 30 block columns, the highest possible triangle would have height equal to seven.

Input

The first line of the input contains an integer N, $1 \leq N \leq 50000$, representing the number of columns in the wall. The second line contains N integers A_i, $1 \leq A_i \leq N$, for $1 \leq i \leq N$, indicating the heights of the columns.

Output

For each test case in the input, your program must produce a single line, containing An integer H,, representing the maximum possible height for a triangle in the end.

Examples

Input	Output
16 5 6 5 8 9 10 5 8 9 5 7 9 9 9 6 3 8 5 1 1 1 1 1 1 3	6 1

Problem J

Olympic Games

A group of investors is thinking on investing heavily in athletes of the Brazilian delegation after the Olympics in Rio. For this, they have watched N athletes and realized that some are in declined and others in rise. Particularly, the group is looking at two attributes on each athlete: fatigue and skill. They will note the scores for these attributes of each athlete at the end of the 2016 olympics. Then the group estimated the rate at which each athlete loses or gains skill and the rate at which each athlete gets tired over time, they realized these rates are constant for both attributes.

Those who bet realize that the noted data allow you to define what they decided to call the golden athlete: an athlete who, in a given period of time is the less tired and the more skilled. It was decided that investments will be made only on golden athletes. Find out how many athletes from those who were observed will receive an investment. Consider that time $t = 0$ is the time of the Rio Olympics: no athlete that become golden before this time will be invested. Also consider that any time after the Rio Olympics should be considered regardles how large it is. An athlete who is golden athlete at time $t = 0$ will be invested.

Input

The input contains several test cases. In each test case, Input starts with a single number N ($1 \leq N \leq 10^5$), the number of athletes. Next N lines, each with 4 integer numbers: M_i, H_t, C_i, C_t ($-10^6 < H_i, H_t, C_i, C_t \leq 10^6$, $H_t, C_t \neq 0$): The initial skill ot i-th athlete, it's variation rate of skill, the initial fatigue and the variation rate of fatigue.

Output

For each test case in the input, Print a single line with the number of athletes that will receive an investment from the group.

Examples

Input	Output
3 3 2 1 2 2 2 2 2 1 2 3 2 6 1 10 5 8 8 7 12 -5 10 -2 -3 8 -3 -5 -8 -12 0 1 10 2 8 3 9 -3	1 0

Problem K

Shrinking polygons kit

A *Shrinking polygons kit* is a tool used a lot in geometric magic classes at Nlogonia. The *kit* consists of two points, A and B in the cartesian plane. Consider a convex polygon vertices given by $1, 2...N$, in that order. To shrink this polygon using the *kit*, some rules should be followed. Each vertex x of the polygon should be moved once: To the midpoint of the segment Ax or the midpoint of the segment Bx. The shrinking procedure should produce a new convex polygon that preserves the order of the vertex on the original polygon. In other words, considering all possible ways to apply the kit, only those where the final sequence of vertices is $1, 2...N$ representing a convex polygon are valid. Note the original convex polygon vertices can be on clockwise order and a valid shrink operation can produce a convex polygon in counter-clockwise vertices order. Only the relative order of the points is important, not the direction.

It is known that geometrical magic is not the forte of most students. The teacher asked them to use the *kit* to shrink a convex polygon to obtain the smallest possible area. A friend has begged you to solve the problem for him. Answer the smallest possible are of the given polygon.

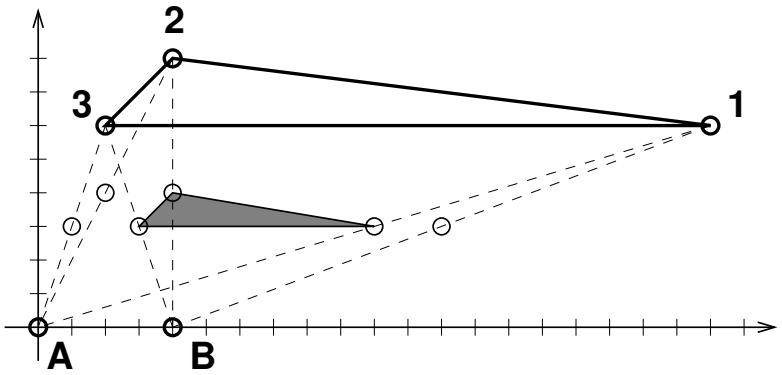


Figure above shows a valid use of the *kit*, where the shaded polygon is the smallest possible area that preserves the sequence of vertices. Points A and B correspond to the points of the *kit*. Note that, despite the name *shrink* sometimes you can use the *kit* to increase the area of the polygon! As geometry is hard!.

Note that a single point or a line are not considered polygons. Thus, if a use of the *kit* produces as a result something other than a convex polygon then it is not a valid use of the *kit*.

Input

The input contains several test cases. In each test case, The first line of input contains an integer number N ($3 \leq N \leq 10^5$), the number of vertices in the polygon. Next N lines, each contains two integer numbers x, y ($-10^6 \leq x, y \leq 10^6$), the polygon vertices. The list line of input contains four integer numbers, A_x, A_y, B_x e B_y ($-10^6 \leq A_x, A_y, B_x, B_y \leq 10^6$), the coordinates x and y for A and the coordinates x and y for B, respectively. The points on the input will be given in the order they are on the polygon, clockwise or anti-clockwise way. There will be no repeated points on the convex polygon.

Output

For each test case in the input, Print a real number with 3 decimal places of precision, representing the smallest area possible for a polygon obtained using the *kit*.

Examples

Input	Output
3 20 6 4 8 2 6 0 0 4 0 3 0 4 4 4 0 0 3 -2 -3 -2 3 0 4 4 4 0 0 2 -2 -2 -2	3.500 1.000 2.000

Problem L
Tiles

Avelino has a mosaic on one wall of his home. It is a very old mosaic, composed of small colored tiles. As it is old some tiles fell off over the years forming holes.

Avelino wants to restore the mosaic covering the holes with new tiles. However, to save some money, Avelino wants to buy tiles of a single color to fill the holes. He wants to buy tiles of the original color or a color that is not in the mosaic.

Being a mosaic, it is not desired to have large areas with the same color. Avelino decided that he will choose the color of the tiles trying to make the size of the smallest monochromatic area as small as possible, so it has more details. Note that there may be more than a possible color. A monochromatic area is an area that all the tiles in it are of the same color. Two adjacent tiles belonging to the same area will have the same color, two tiles are adjacent if they share one side.

See the first example, there are three areas of color 1 (one of size 3 and two of size 2), one area of color 2 (size 3) and one area of color 3 (size 7). The possible answer would be to choose color 2, this way the smallest monochromatic area size is 2. If we choose color 1 the smallest monochromatic area would be 3.

Create a program that prints the size of the smallest possible monochromatic area.

Input

The first line of input contains two numbers H and L, the height and length of the mosaic, respectively $1 \leq H \leq 200$ and $1 \leq L \leq 200$. Next H lines contains each L numbers, separated by a wihte space, the color of each tile. A value of 0 represents a hole in the mosaic and a value $i \neq 0$ represents a tile with i color $1 \leq i \leq 40000$

Output

Your program should pring a single line with an integer representing the size of the smallest possible monochromatic area.

Examples

Input	Output
3 8 3 3 3 1 1 0 0 0 3 1 1 0 2 2 0 1 3 3 3 0 0 2 1 1 3 7 1 1 0 2 2 1 1 1 1 0 2 2 1 1 1 1 0 0 3 3 3 3 6 2 2 2 2 0 2 2 2 2 0 2 2 2 2 2 2 0 2	2 3 1