Codeforces Round #551 (Div. 2)

A. Serval and Bus

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

It is raining heavily. But this is the first day for Serval, who just became 3 years old, to go to the kindergarten. Unfortunately, he lives far from kindergarten, and his father is too busy to drive him there. The only choice for this poor little boy is to wait for a bus on this rainy day. Under such circumstances, the poor boy will use the first bus he sees no matter where it goes. If several buses come at the same time, he will choose one randomly.

Serval will go to the bus station at time t, and there are n bus routes which stop at this station. For the i-th bus route, the first bus arrives at time s_i minutes, and each bus of this route comes d_i minutes later than the previous one.

As Serval's best friend, you wonder which bus route will he get on. If several buses arrive at the same time, you can print any of them.

Input

The first line contains two space-separated integers n and t ($1 \le n \le 100$, $1 \le t \le 10^5$) — the number of bus routes and the time Serval goes to the station.

Each of the next n lines contains two space-separated integers s_i and d_i ($1 \le s_i, d_i \le 10^5$) — the time when the first bus of this route arrives and the interval between two buses of this route.

Output

Print one number — what bus route Serval will use. If there are several possible answers, you can print any of them.

Examples

input
2 2 6 4 9 5
output
1

input
5 5 3 3 2 5 5 6 4 9 6 1
output
3

input
3 7 2 2 2 3 2 4
output
1

Note

In the first example, the first bus of the first route arrives at time 6, and the first bus of the second route arrives at time 9, so the first route is the answer.

In the second example, a bus of the third route arrives at time 5, so it is the answer.

In the third example, buses of the first route come at times 2, 4, 6, 8, and so fourth, buses of the second route come at times 2, 5, 8, and so fourth and buses of the third route come at times 2, 6, 10, and so on, so 1 and 2 are both acceptable answers while 3 is not.

B. Serval and Toy Bricks

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

Luckily, Serval got onto the right bus, and he came to the kindergarten on time. After coming to kindergarten, he found the toy bricks very funny.

He has a special interest to create difficult problems for others to solve. This time, with many $1 \times 1 \times 1$ toy bricks, he builds up a 3-dimensional object. We can describe this object with a $n \times m$ matrix, such that in each cell (i, j), there are $h_{i,j}$ bricks standing on the top of each other.

However, Serval doesn't give you any $h_{i,j}$, and just give you the front view, left view, and the top view of this object, and he is now asking you to restore the object. Note that in the front view, there are m columns, and in the i-th of them, the height is the maximum of $h_{1,i}, h_{2,i}, \ldots, h_{n,i}$. It is similar for the left view, where there are n columns. And in the top view, there is an $n \times m$ matrix $t_{i,j}$, where $t_{i,j}$ is 0 or 1. If $t_{i,j}$ equals 1, that means $h_{i,j} > 0$, otherwise, $h_{i,j} = 0$.

However, Serval is very lonely because others are bored about his unsolvable problems before, and refused to solve this one, although this time he promises there will be at least one object satisfying all the views. As his best friend, can you have a try?

Input

The first line contains three positive space-separated integers n, m, h ($1 \leq n, m, h \leq 100$) — the length, width and height.

The second line contains m non-negative space-separated integers $a_1, a_2, \ldots, a_m$, where a_i is the height in the i-th column from left to right of the front view ($0 \leq a_i \leq h$).

The third line contains n non-negative space-separated integers $b_1, b_2, \ldots, b_n$ ($0 \leq b_j \leq h$), where b_j is the height in the j-th column from left to right of the left view.

Each of the following n lines contains m numbers, each is 0 or 1, representing the top view, where j-th number of i-th row is 1 if $h_{i,j} > 0$, and 0 otherwise.

It is guaranteed that there is at least one structure satisfying the input.

Output

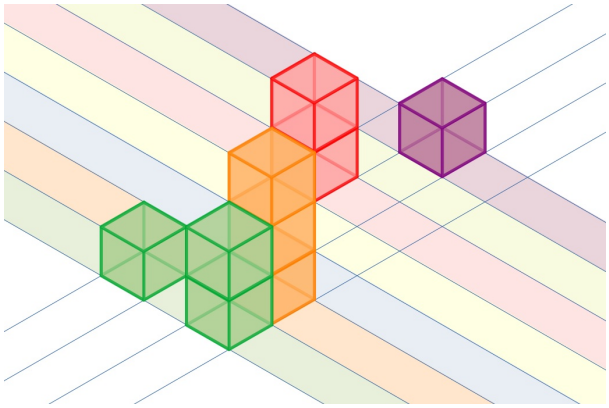
Output n lines, each of them contains m integers, the j-th number in the i-th line should be equal to the height in the corresponding position of the top view. If there are several objects satisfying the views, output any one of them.

Examples

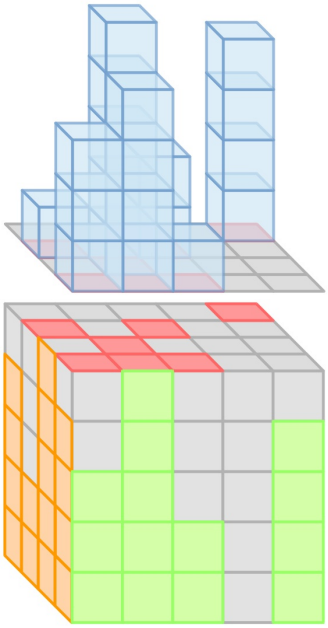
input
3 7 3 2 3 0 0 2 0 1 2 1 3 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
output
1 0 0 0 2 0 0 0 0 0 0 0 0 1 2 3 0 0 0 0 0

input
4 5 5 3 5 2 0 4 4 2 5 4 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 0 0
output
0 0 0 0 4 1 0 2 0 0 0 5 0 0 0 3 4 1 0 0

Note



The graph above illustrates the object in the first example.



The first graph illustrates the object in the example output for the second example, and the second graph shows the three-view drawing of it.

C. Serval and Parenthesis Sequence

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

Serval soon said goodbye to Japari kindergarten, and began his life in Japari Primary School.

In his favorite math class, the teacher taught him the following interesting definitions.

A *parenthesis sequence* is a string, containing only characters "(" and ")".

A *correct parenthesis sequence* is a parenthesis sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example, parenthesis sequences "()()", "(())" are correct (the resulting expressions are: "(1+1)+(1+1)", "((1+1)+1)"), while ")(" and ")" are not. Note that the empty string is a correct parenthesis sequence by definition.

We define that $|s|$ as the length of string s. A *strict prefix* $s[1 \ldots l]$ $(1 \leq l < |s|)$ of a string $s = s_1 s_2 \ldots s_{|s|}$ is string $s_1 s_2 \ldots s_l$. Note that the empty string and the whole string are not strict prefixes of any string by the definition.

Having learned these definitions, he comes up with a new problem. He writes down a string s containing only characters "(", ")" and "?". And what he is going to do, is to replace each of the "?" in s independently by one of "(" and ")" to make all strict prefixes of the new sequence not a correct parenthesis sequence, while the new sequence should be a correct parenthesis sequence.

After all, he is just a primary school student so this problem is too hard for him to solve. As his best friend, can you help him to replace the question marks? If there are many solutions, any of them is acceptable.

Input

The first line contains a single integer $|s|$ $(1 \leq |s| \leq 3 \cdot 10^5)$, the length of the string.

The second line contains a string s, containing only "(", ")" and "?".

Output

A single line contains a string representing the answer.

If there are many solutions, any of them is acceptable.

If there is no answer, print a single line containing ":(" (without the quotes).

Examples

input
6 (?????
output
(()())

input
10 (???(???(?
output
:(

Note
It can be proved that there is no solution for the second sample, so print ":(".

D. Serval and Rooted Tree

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

Now Serval is a junior high school student in Japari Middle School, and he is still thrilled on math as before.

As a talented boy in mathematics, he likes to play with numbers. This time, he wants to play with numbers on a rooted tree.

A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. A parent of a node v is the last different from v vertex on the path from the root to the vertex v. Children of vertex v are all nodes for which v is the parent. A vertex is a leaf if it has no children.

The rooted tree Serval owns has n nodes, node 1 is the root. Serval will write some numbers into all nodes of the tree. However, there are some restrictions. Each of the nodes except leaves has an operation $\max$ or $\min$ written in it, indicating that the number in this node should be equal to the maximum or minimum of all the numbers in its sons, respectively.

Assume that there are k leaves in the tree. Serval wants to put integers $1, 2, \ldots, k$ to the k leaves (each number should be used exactly once). He loves large numbers, so he wants to maximize the number in the root. As his best friend, can you help him?

Input
The first line contains an integer n ($2 \leq n \leq 3 \cdot 10^5$), the size of the tree.

The second line contains n integers, the i-th of them represents the operation in the node i. 0 represents $\min$ and 1 represents $\max$. If the node is a leaf, there is still a number of 0 or 1, but you can ignore it.

The third line contains $n-1$ integers $f_2, f_3, \ldots, f_n$ ($1 \leq f_i \leq i-1$), where f_i represents the parent of the node i.

Output
Output one integer — the maximum possible number in the root of the tree.

Examples

input
6 1 0 1 1 0 1 1 2 2 2 2
output
1

input
5 1 0 1 0 1 1 1 1 1
output
4

input
8 1 0 0 1 0 1 1 0 1 1 2 2 3 3 3

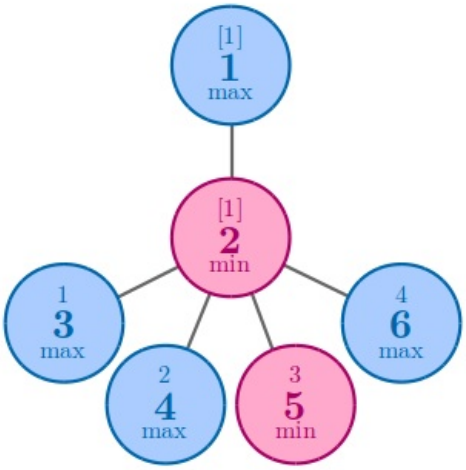
output
4

input
9 1 1 0 0 1 0 1 0 1 1 1 2 2 3 3 4 4
output
5

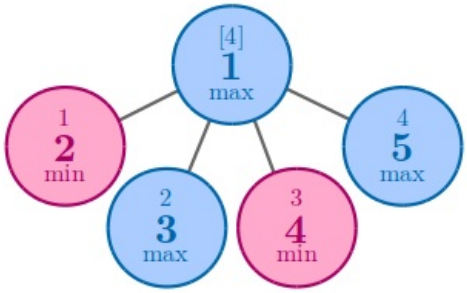
Note

Pictures below explain the examples. The numbers written in the middle of the nodes are their indices, and the numbers written on the top are the numbers written in the nodes.

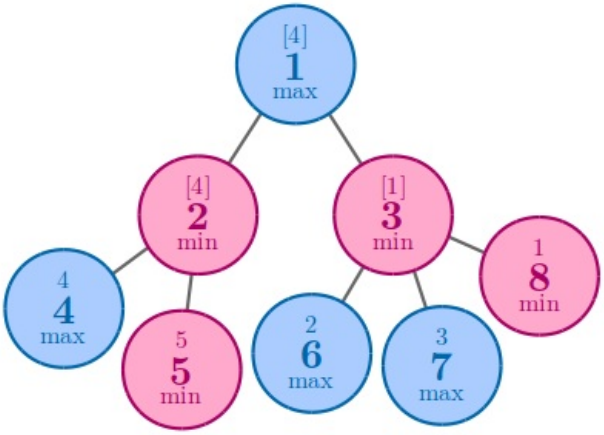
In the first example, no matter how you arrange the numbers, the answer is 1.



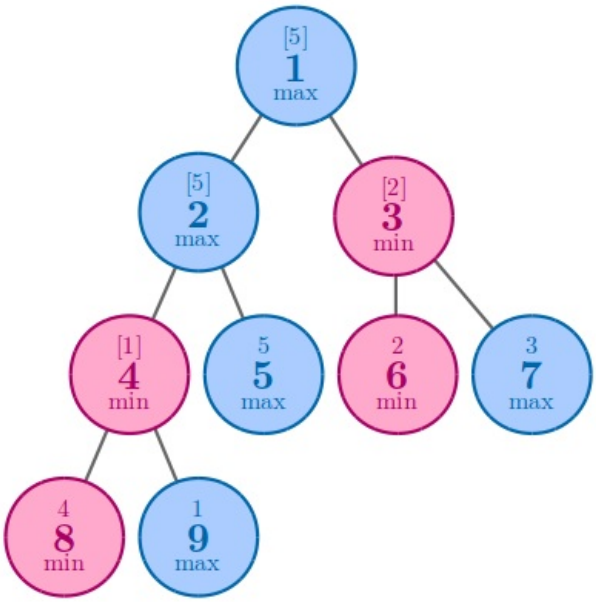
In the second example, no matter how you arrange the numbers, the answer is 4.



In the third example, one of the best solution to achieve 4 is to arrange 4 and 5 to nodes 4 and 5.



In the fourth example, the best solution is to arrange 5 to node 5.



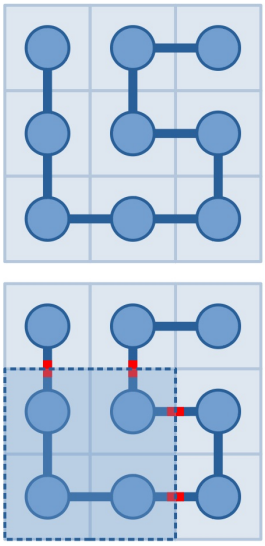
E. Serval and Snake

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

This is an interactive problem.

Now Serval is a senior high school student in Japari Middle School. However, on the way to the school, he must go across a pond, in which there is a dangerous snake. The pond can be represented as a $n \times n$ grid. The snake has a head and a tail in different cells, and its body is a series of adjacent cells connecting the head and the tail without self-intersecting. If Serval hits its head or tail, the snake will bite him and he will die.

Luckily, he has a special device which can answer the following question: you can pick a rectangle, it will tell you the number of times one needs to cross the border of the rectangle walking cell by cell along the snake from the head to the tail. The pictures below show a possible snake and a possible query to it, which will get an answer of 4.



Today Serval got up too late and only have time to make 2019 queries. As his best friend, can you help him find the positions of the head and the tail?

Note that two cells are adjacent if and only if they have a common edge in the grid, and a snake can have a body of length 0, that means it only has adjacent head and tail.

Also note that the snake is sleeping, so it won't move while Serval using his device. And what's obvious is that the snake position does not depend on your queries.

Input

The first line contains a single integer n ($2 \leq n \leq 1000$) — the size of the grid.

Output

When you are ready to answer, you should print ! x1 y1 x2 y2, where (x_1, y_1) represents the position of the head and (x_2, y_2) represents the position of the tail. You can print head and tail in any order.

Interaction

To make a query, you should print `? x1 y1 x2 y2` ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$), representing a rectangle consisting of all cells (x, y) such that $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$. You will get a single integer as the answer.

After printing a query, do not forget to output the end of line and flush the output, otherwise you will get `Idleness limit exceeded`. To do this, use:

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

Answer -1 instead of a valid answer means that you made an invalid query or exceeded the maximum number of queries. Exit immediately after receiving -1 and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

If your program cannot find out the head and tail of the snake correctly, you will also get a `Wrong Answer` verdict.

Hacks

To make a hack, print a single integer n ($2 \le n \le 1000$) in the first line, indicating the size of the grid.

Then print an integer k ($2 \le k \le n^2$) in the second line, indicating the length of the snake.

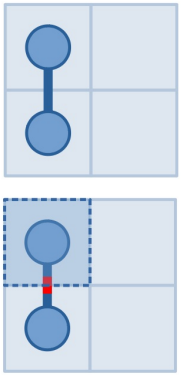
In the next k lines, print k pairs of integers x_i, y_i ($1 \le x_i, y_i \le n$), each pair in a single line, indicating the i-th cell of snake, such that the adjacent pairs are adjacent, and all k pairs are distinct.

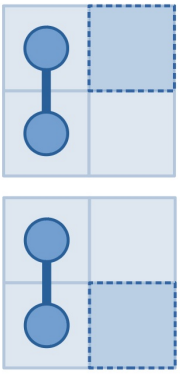
Examples

input
2 1 0 0
output
? 1 1 1 1 ? 1 2 1 2 ? 2 2 2 2 ! 1 1 2 1

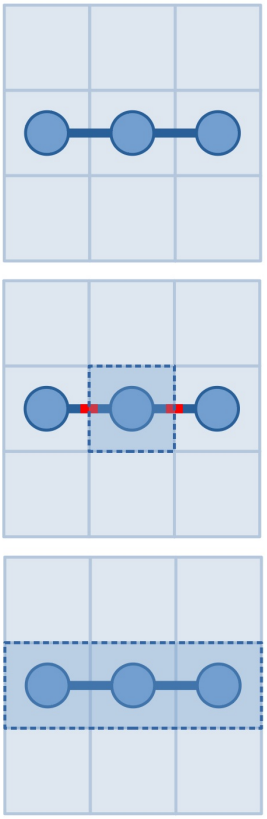
input
3 2 0
output
? 2 2 2 2 ? 2 1 2 3 ! 2 1 2 3

Note





The pictures above show our queries and the answers in the first example. We first made a query for $(1, 1)$ and got an answer 1, then found that it must be connected to exactly one other cell. Then we made a query for $(1, 2)$ and got an answer of 0, then knew that the snake never entered it. So the cell connected to $(1, 1)$ must be $(2, 1)$. Then we made a query for $(2, 2)$ and got an answer 0 , then knew that it never entered $(2, 2)$ as well. So the snake cannot leave $(2, 1)$, which implies that the answer is $(1, 1)$ and $(2, 1)$.



The pictures above show our queries and the answers in the second example. By making query to $(2, 2)$ and receiving 2, we found that the snake occupies $(2, 2)$. And by making query to rectangle from $(2, 1)$ to $(2, 3)$ and receiving answer 0, we knew that it never goes out of the rectangle from $(2, 1)$ to $(2, 3)$. Since the first answer is 2, both $(2, 1)$ and $(2, 3)$ must be occupied but none of others, so the answer is $(2, 1)$ and $(2, 3)$.

F. Serval and Bonus Problem

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

Getting closer and closer to a mathematician, Serval becomes a university student on math major in Japari University. On the Calculus class, his teacher taught him how to calculate the expected length of a random subsegment of a given segment. Then he left a bonus problem as homework, with the award of a garage kit from IOI. The bonus is to extend this problem to the general case as follows.

You are given a segment with length l. We randomly choose n segments by choosing two points (maybe with non-integer coordinates) from the given segment equiprobably and the interval between the two points forms a segment. You are given the number of random segments n, and another integer k. The $2n$ endpoints of the chosen segments split the segment into $(2n + 1)$ intervals. Your task is to calculate the expected total length of those intervals that are covered by at least k segments of the n random segments.

You should find the answer modulo 998244353.

Input

First line contains three space-separated positive integers n, k and l ($1 \le k \le n \le 2000$, $1 \le l \le 10^9$).

Output

Output one integer — the expected total length of all the intervals covered by at least k segments of the n random segments modulo 998244353.

Formally, let $M = 998244353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, 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 $x \cdot q \equiv p \pmod{M}$.

Examples

input
1 1 1
output
332748118

input
6 2 1
output
760234711

input
7 5 3
output
223383352

input
97 31 9984524
output
267137618

Note

In the first example, the expected total length is $\int_0^1 \int_0^1 |x - y| \, \mathrm{d}x \, \mathrm{d}y = \frac{1}{3}$, and 3^{-1} modulo 998244353 is 332748118.