Codeforces Round #488 by NEAR (Div. 1)

A. Two Squares

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

You are given two squares, one with sides parallel to the coordinate axes, and another one with sides at 45 degrees to the coordinate axes. Find whether the two squares intersect.

The interior of the square is considered to be part of the square, i.e. if one square is completely inside another, they intersect. If the two squares only share one common point, they are also considered to intersect.

Input

The input data consists of two lines, one for each square, both containing 4 pairs of integers. Each pair represents coordinates of one vertex of the square. Coordinates within each line are either in clockwise or counterclockwise order.

The first line contains the coordinates of the square with sides parallel to the coordinate axes, the second line contains the coordinates of the square at 45 degrees.

All the values are integer and between $$$-100$$$ and $$$100$$$.

Output

Print "`Yes`" if squares intersect, otherwise print "`No`".

You can print each letter in any case (upper or lower).

Examples

input
0 0 6 0 6 6 0 6 1 3 3 5 5 3 3 1
output
YES

input
0 0 6 0 6 6 0 6 7 3 9 5 11 3 9 1
output
NO

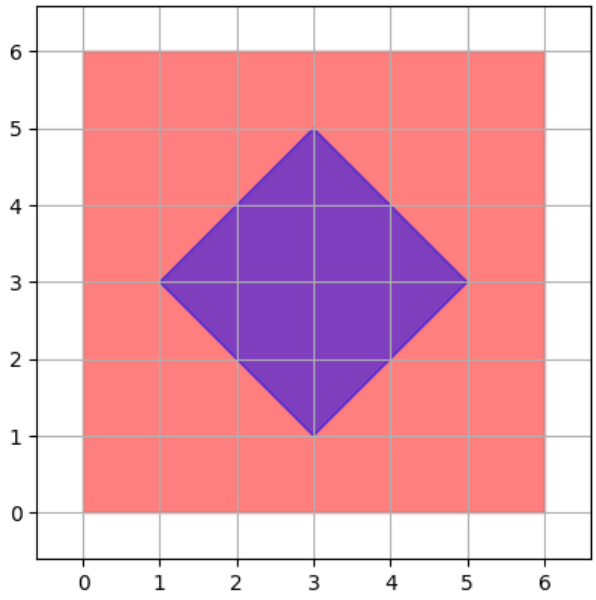
input
6 0 6 6 0 6 0 0 7 4 4 7 7 10 10 7
output
YES

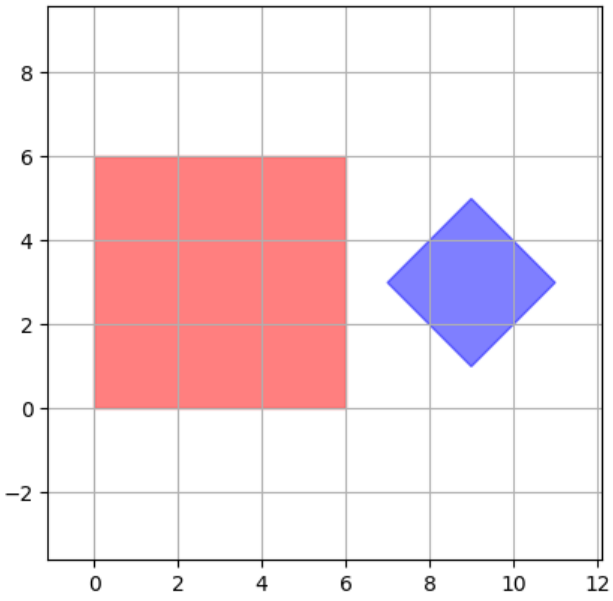
Note

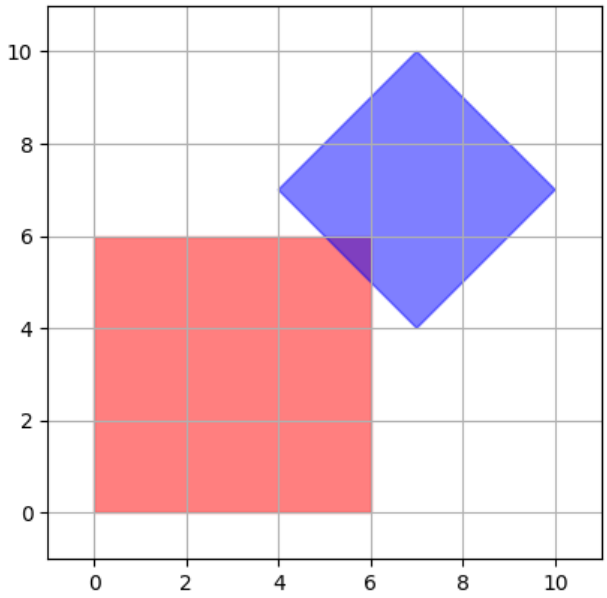
In the first example the second square lies entirely within the first square, so they do intersect.

In the second sample squares do not have any points in common.

Here are images corresponding to the samples:


6
5
4
3
2
1
0
0
1
2
3
4
5
6


8
6
4
2
0
−2
0
2
4
6
8
10
12



B. Open Communication

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

Two participants are each given a pair of distinct numbers from 1 to 9 such that there's exactly one number that is present in both pairs. They want to figure out the number that matches by using a communication channel you have access to without revealing it to you.

Both participants communicated to each other a set of pairs of numbers, that includes the pair given to them. Each pair in the communicated sets comprises two different numbers.

Determine if you can with certainty deduce the common number, or if you can determine with certainty that both participants know the number but you do not.

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$1 \le n, m \le 12$$$) — the number of pairs the first participant communicated to the second and vice versa.

The second line contains $$$n$$$ pairs of integers, each between $$$1$$$ and $$$9$$$, — pairs of numbers communicated from first participant to the second.

The third line contains $$$m$$$ pairs of integers, each between $$$1$$$ and $$$9$$$, — pairs of numbers communicated from the second participant to the first.

All pairs within each set are distinct (in particular, if there is a pair $$$(1,2)$$$, there will be no pair $$$(2,1)$$$ within the same set), and no pair contains the same number twice.

It is guaranteed that the two sets do not contradict the statements, in other words, there is pair from the first set and a pair from the second set that share exactly one number.

Output

If you can deduce the shared number with certainty, print that number.

If you can with certainty deduce that both participants know the shared number, but you do not know it, print $$$0$$$.

Otherwise print $$$-1$$$.

Examples

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

input

```
2 2
1 2 3 4
1 5 6 4
```

output

```
0
```

input

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

output

```
-1
```

Note

In the first example the first participant communicated pairs $$$(1,2)$$$ and $$$(3,4)$$$, and the second communicated $$$(1,5)$$$, $$$(3,4)$$$. Since we know that the actual pairs they received share exactly one number, it can't be that they both have $$$(3,4)$$$. Thus, the first participant has $$$(1,2)$$$ and the second has $$$(1,5)$$$, and at this point you already know the shared number is $$$1$$$.

In the second example either the first participant has $$$(1,2)$$$ and the second has $$$(1,5)$$$, or the first has $$$(3,4)$$$ and the second has $$$(6,4)$$$. In the first case both of them know the shared number is $$$1$$$, in the second case both of them know the shared number is $$$4$$$. You don't have enough information to tell $$$1$$$ and $$$4$$$ apart.

In the third case if the first participant was given $$$(1,2)$$$, they don't know what the shared number is, since from their perspective the second participant might have been given either $$$(1,3)$$$, in which case the shared number is $$$1$$$, or $$$(2,3)$$$, in which case the shared number is $$$2$$$. While the second participant does know the number with certainty, neither you nor the first participant do, so the output is $$$-1$$$.

C. Careful Maneuvering

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

There are two small spaceship, surrounded by two groups of enemy larger spaceships. The space is a two-dimensional plane, and one group of the enemy spaceships is positioned in such a way that they all have integer $$$y$$$-coordinates, and their $$$x$$$-coordinate is equal to $$$-100$$$, while the second group is positioned in such a way that they all have integer $$$y$$$-coordinates, and their $$$x$$$-coordinate is equal to $$$100$$$.

Each spaceship in both groups will simultaneously shoot two laser shots (infinite ray that destroys any spaceship it touches), one towards each of the small spaceships, all at the same time. The small spaceships will be able to avoid all the laser shots, and now want to position themselves at some locations with $$$x=0$$$ (with not necessarily integer $$$y$$$-coordinates), such that the rays shot at them would destroy as many of the enemy spaceships as possible. Find the largest numbers of spaceships that can be destroyed this way, assuming that the enemy spaceships can't avoid laser shots.

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$1 \le n, m \le 60$$$), the number of enemy spaceships with $$$x = -100$$$ and the number of enemy spaceships with $$$x = 100$$$, respectively.

The second line contains $$$n$$$ integers $$$y_{1,1}, y_{1,2}, \ldots, y_{1,n}$$$ ($$$|y_{1,i}| \le 10\,000$$$) — the $$$y$$$-coordinates of the spaceships in the first group.

The third line contains $$$m$$$ integers $$$y_{2,1}, y_{2,2}, \ldots, y_{2,m}$$$ ($$$|y_{2,i}| \le 10\,000$$$) — the $$$y$$$-coordinates of the spaceships in the second group.

The $$$y$$$ coordinates are not guaranteed to be unique, even within a group.

Output

Print a single integer – the largest number of enemy spaceships that can be destroyed.

Examples

input

```
3 9
1 2 3
1 2 3 7 8 9 11 12 13
```

output

```
9
```

input

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

output
10

Note
In the first example the first spaceship can be positioned at $$$(0, 2)$$$, and the second – at $$$(0, 7)$$$. This way all the enemy spaceships in the first group and $$$6$$$ out of $$$9$$$ spaceships in the second group will be destroyed.

In the second example the first spaceship can be positioned at $$$(0, 3)$$$, and the second can be positioned anywhere, it will be sufficient to destroy all the enemy spaceships.

D. Compute Power

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

You need to execute several tasks, each associated with number of processors it needs, and the compute power it will consume.

You have sufficient number of analog computers, each with enough processors for any task. Each computer can execute up to one task at a time, and no more than two tasks total. The first task can be any, the second task on each computer must use strictly less power than the first. You will assign between 1 and 2 tasks to each computer. You will then first execute the first task on each computer, wait for all of them to complete, and then execute the second task on each computer that has two tasks assigned.

If the average compute power per utilized processor (the sum of all consumed powers for all tasks presently running divided by the number of utilized processors) across all computers exceeds some unknown threshold during the execution of the first tasks, the entire system will blow up. There is no restriction on the second tasks execution. Find the lowest threshold for which it is possible.

Due to the specifics of the task, you need to print the answer multiplied by 1000 and rounded up.

Input
The first line contains a single integer n ($1 \le n \le 50$) — the number of tasks.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^8$), where a_i represents the amount of power required for the i-th task.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 100$), where b_i is the number of processors that i-th task will utilize.

Output
Print a single integer value — the lowest threshold for which it is possible to assign all tasks in such a way that the system will not blow up after the first round of computation, multiplied by 1000 and rounded up.

Examples

input
6 8 10 9 9 8 10 1 1 1 1 1 1
output
9000

input
6 8 10 9 9 8 10 1 10 5 5 1 10
output
1160

Note
In the first example the best strategy is to run each task on a separate computer, getting average compute per processor during the first round equal to 9.

In the second task it is best to run tasks with compute 10 and 9 on one computer, tasks with compute 10 and 8 on another, and tasks with compute 9 and 8 on the last, averaging (10 + 10 + 9) / (10 + 10 + 5) = 1.16 compute power per processor during the first round.

E. Nikita and Order Statistics

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

Nikita likes tasks on order statistics, for example, he can easily find the $$$k$$$-th number in increasing order on a segment of an array. But now Nikita wonders how many segments of an array there are such that a given number $$$x$$$ is the $$$k$$$-th number in increasing order on this

segment. In other words, you should find the number of segments of a given array such that there are exactly $$$k$$$ numbers of this segment which are less than $$$x$$$.

Nikita wants to get answer for this question for each $$$k$$$ from $$$0$$$ to $$$n$$$, where $$$n$$$ is the size of the array.

Input
The first line contains two integers $$$n$$$ and $$$x$$$ $$$(1 \le n \le 2 \cdot 10^5, -10^9 \le x \le 10^9)$$$.

The second line contains $$$n$$$ integers $$$a_1, a_2, \ldots, a_n$$$ $$$(-10^9 \le a_i \le 10^9)$$$ — the given array.

Output
Print $$$n+1$$$ integers, where the $$$i$$$-th number is the answer for Nikita's question for $$$k=i-1$$$.

Examples

input
5 3 1 2 3 4 5
output
6 5 4 0 0 0

input
2 6 -5 9
output
1 2 0

input
6 99 -1 -1 -1 -1 -1 -1
output
0 6 5 4 3 2 1

F. The Moral Dilemma

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

Hibiki and Dita are in love with each other, but belong to communities that are in a long lasting conflict. Hibiki is deeply concerned with the state of affairs, and wants to figure out if his relationship with Dita is an act of love or an act of treason.

Hibiki prepared several binary features his decision will depend on, and built a three layer logical circuit on top of them, each layer consisting of one or more logic gates. Each gate in the circuit is either "`or`", "`and`", "`nor`" (not or) or "`nand`" (not and). Each gate in the first layer is connected to exactly two features. Each gate in the second layer is connected to exactly two gates in the first layer. The third layer has only one "`or`" gate, which is connected to all the gates in the second layer (in other words, the entire circuit produces 1 if and only if at least one gate in the second layer produces 1).

The problem is, Hibiki knows very well that when the person is in love, his ability to think logically degrades drastically. In particular, it is well known that when a person in love evaluates a logical circuit in his mind, every gate evaluates to a value that is the opposite of what it was supposed to evaluate to. For example, "`or`" gates return 1 if and only if both inputs are zero, "t{nand}" gates produce 1 if and only if both inputs are one etc.

In particular, the "`or`" gate in the last layer also produces opposite results, and as such if Hibiki is in love, the entire circuit produces 1 if and only if all the gates on the second layer produced 0.

Hibiki can't allow love to affect his decision. He wants to know what is the smallest number of gates that needs to be removed from the second layer so that the output of the circuit for all possible inputs doesn't depend on whether Hibiki is in love or not.

Input
The first line contains three integers $$$n$$$, $$$m$$$, $$$k$$$ ($$$2 \le n, m \le 50$$$; $$$1 \le k \le 50$$$) — the number of input features, the number of gates in the first layer, and the number of gates in the second layer correspondingly.

The second line contains $$$m$$$ pairs of strings separated by spaces describing the first layer. The first string in each pair describes the gate ("`and`", "`or`", "`nand`" or "`nor`"), and the second string describes the two input features the gate is connected two as a string consisting of exactly $$$n$$$ characters, with exactly two characters (that correspond to the input features the gate is connected to) equal to '`x`' and the remaining characters equal to "`.`".

The third line contains $$$k$$$ pairs of strings separated by spaces describing the second layer in the same format, where the strings that describe the input parameters have length $$$m$$$ and correspond to the gates of the first layer.

Output
Print the number of gates that need to be removed from the second layer so that the output of the remaining circuit doesn't depend on whether Hibiki is in love or not.

If no matter how many gates are removed the output of the circuit continues to depend on Hibiki's feelings, print $$$-1$$$.

Examples

input
2 2 2 and xx nand xx and xx or xx
output
1

input
3 2 2 and xx. nor .xx and xx nor xx
output
-1

input
4 4 5 nor x..x and ..xx and xx.. nand xx.. nand ..xx nor ..xx and xx.. nor ..xx or ..xx
output
2

Note
In the first example the two gates in the first layer are connected to the same inputs, but first computes "`and`" while second computes "`nand`", and as such their output is always different no matter what the input is and whether Hibiki is in love or not. The second layer has "`or`" and "`and`" gates both connected to the two gates in the first layer. If Hibiki is not in love, the "`and`" gate will produce 0 and the "`or`" gate will produce 1 no matter what input features are equal to, with the final "`or`" gate in the third layer always producing the final answer of 1. If Hibiki is in love, "`and`" gate in the second layer will produce 1 and "`or`" gate will produce 0 no matter what the input is, with the final "`or`" gate in the third layer producing the final answer of 0. Thus, if both gates in the second layer are kept, the output of the circuit does depend on whether Hibiki is in love. If any of the two gates in the second layer is dropped, the output of the circuit will no longer depend on whether Hibiki is in love or not, and hence the answer is 1.

In the second example no matter what gates are left in the second layer, the output of the circuit will depend on whether Hibiki is in love or not.

In the third example if Hibiki keeps second, third and fourth gates in the second layer, the circuit will not depend on whether Hibiki is in love or not. Alternatively, he can keep the first and the last gates. The former requires removing two gates, the latter requires removing three gates, so the former is better, and the answer is 2.