Codeforces Round #488 by NEAR (Div. 2)

A. Fingerprints

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

You are locked in a room with a door that has a keypad with 10 keys corresponding to digits from 0 to 9. To escape from the room, you need to enter a correct code. You also have a sequence of digits.

Some keys on the keypad have fingerprints. You believe the correct code is the longest not necessarily contiguous subsequence of the sequence you have that only contains digits with fingerprints on the corresponding keys. Find such code.

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$1 \le n, m \le 10$$$) representing the number of digits in the sequence you have and the number of keys on the keypad that have fingerprints.

The next line contains $$$n$$$ distinct space-separated integers $$$x_1, x_2, \ldots, x_n$$$ ($$$0 \le x_i \le 9$$$) representing the sequence.

The next line contains $$$m$$$ distinct space-separated integers $$$y_1, y_2, \ldots, y_m$$$ ($$$0 \le y_i \le 9$$$) — the keys with fingerprints.

Output

In a single line print a space-separated sequence of integers representing the code. If the resulting sequence is empty, both printing nothing and printing a single line break is acceptable.

Examples

input
7 3 3 5 7 1 6 2 8 1 2 7

output
7 1 2

input
4 4 3 4 1 0 0 1 7 9

output
1 0

Note

In the first example, the only digits with fingerprints are $$$1$$$, $$$2$$$ and $$$7$$$. All three of them appear in the sequence you know, $$$7$$$ first, then $$$1$$$ and then $$$2$$$. Therefore the output is `7 1 2`. Note that the order is important, and shall be the same as the order in the original sequence.

In the second example digits $$$0$$$, $$$1$$$, $$$7$$$ and $$$9$$$ have fingerprints, however only $$$0$$$ and $$$1$$$ appear in the original sequence. $$$1$$$ appears earlier, so the output is `1 0`. Again, the order is important.

B. Knights of a Polygonal Table

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

Unlike Knights of a Round Table, Knights of a Polygonal Table deprived of nobility and happy to kill each other. But each knight has some power and a knight can kill another knight if and only if his power is greater than the power of victim. However, even such a knight will torment his conscience, so he can kill no more than $$$k$$$ other knights. Also, each knight has some number of coins. After a kill, a knight can pick up all victim's coins.

Now each knight ponders: how many coins he can have if only he kills other knights?

You should answer this question for each knight.

Input

The first line contains two integers $$$n$$$ and $$$k$$$ $$$(1 \le n \le 10^5, 0 \le k \le \min(n-1,10))$$$ — the number of knights and the number $$$k$$$ from the statement.

The second line contains $$$n$$$ integers $$$p_1, p_2 ,\ldots,p_n$$$ $$$(1 \le p_i \le 10^9)$$$ — powers of the knights. All $$$p_i$$$ are distinct.

The third line contains $$$n$$$ integers $$$c_1, c_2 ,\ldots,c_n$$$ $$$(0 \le c_i \le 10^9)$$$ — the number of coins each knight has.

Output
Print $$$n$$$ integers — the maximum number of coins each knight can have it only he kills other knights.

Examples

input
4 2 4 5 9 7 1 2 11 33
output
1 3 46 36

input
5 1 1 2 3 4 5 1 2 3 4 5
output
1 3 5 7 9

input
1 0 2 3
output
3

Note
Consider the first example.

- The first knight is the weakest, so he can't kill anyone. That leaves him with the only coin he initially has.
- The second knight can kill the first knight and add his coin to his own two.
- The third knight is the strongest, but he can't kill more than $$$k = 2$$$ other knights. It is optimal to kill the second and the fourth knights: $$$2+11+33 = 46$$$.
- The fourth knight should kill the first and the second knights: $$$33+1+2 = 36$$$.

In the second example the first knight can't kill anyone, while all the others should kill the one with the index less by one than their own.

In the third example there is only one knight, so he can't kill anyone.

C. 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

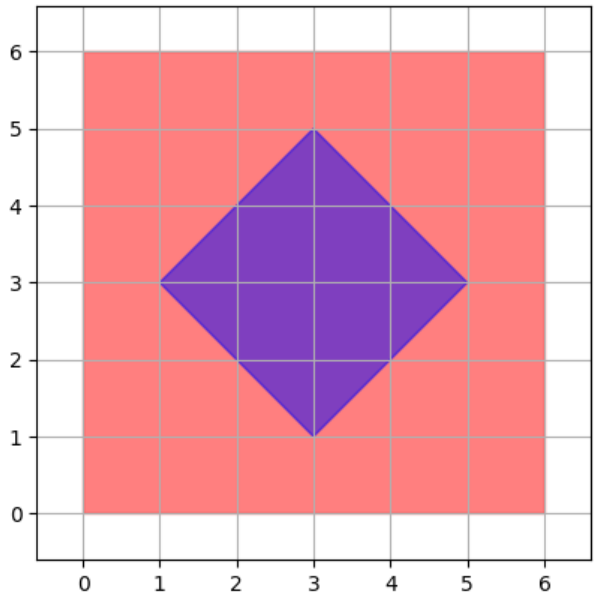
```
YES
```

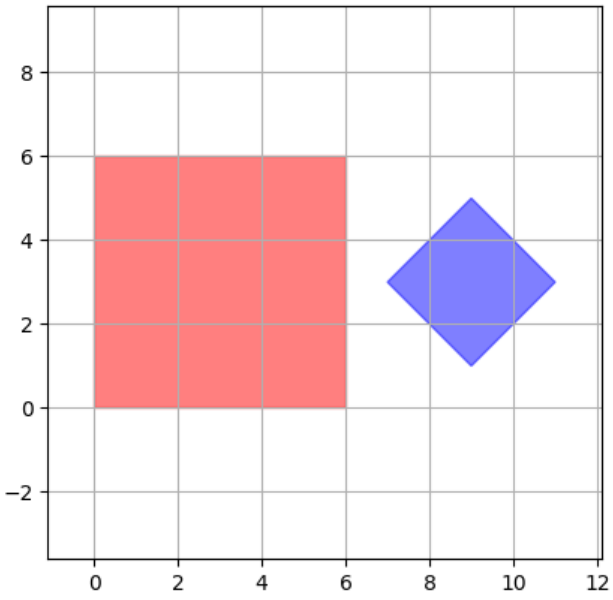
Note

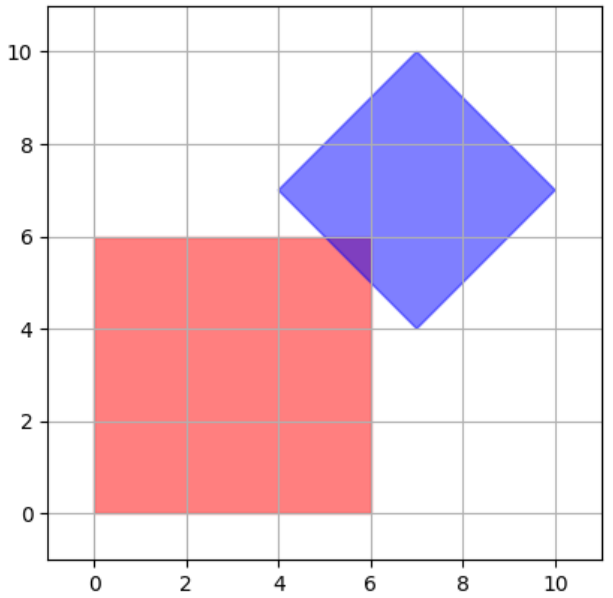
In the first example the second square lies entirely within the first square, so they do intersect.

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

Here are images corresponding to the samples:







D. 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 $$$-1$$$.

E. 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.

F. 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.