A. Currency System in Geraldion

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

A magic island Geraldion, where Gerald lives, has its own currency system. It uses banknotes of several values. But the problem is, the system is not perfect and sometimes it happens that Geraldionians cannot express a certain sum of money with any set of banknotes. Of course, they can use any number of banknotes of each value. Such sum is called *unfortunate*. Gerald wondered: what is the minimum *unfortunate* sum?

Input

The first line contains number n ($1 \le n \le 1000$) — the number of values of the banknotes that used in Geraldion.

The second line contains n distinct space-separated numbers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$) — the values of the banknotes.

Output

Print a single line — the minimum *unfortunate* sum. If there are no unfortunate sums, print -1.

Examples

input

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

output

```
-1
```

B. Gerald is into Art

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

Gerald bought two very rare paintings at the Sotheby's auction and he now wants to hang them on the wall. For that he bought a special board to attach it to the wall and place the paintings on the board. The board has shape of an $a_1 \times b_1$ rectangle, the paintings have shape of a $a_2 \times b_2$ and $a_3 \times b_3$ rectangles.

Since the paintings are painted in the style of abstract art, it does not matter exactly how they will be rotated, but still, one side of both the board, and each of the paintings must be parallel to the floor. The paintings can touch each other and the edges of the board, but can not overlap or go beyond the edge of the board. Gerald asks whether it is possible to place the paintings on the board, or is the board he bought not large enough?

Input

The first line contains two space-separated numbers a_1 and b_1 — the sides of the board. Next two lines contain numbers a_2, b_2, a_3 and b_3 — the sides of the paintings. All numbers a_i, b_i in the input are integers and fit into the range from 1 to 1000.

Output

If the paintings can be placed on the wall, print "YES" (without the quotes), and if they cannot, print "NO" (without the quotes).

Examples

input

```
3 2
1 3
2 1
```

output

```
YES
```

input

```
5 5
3 3
3 3
```

output

```
NO
```

input

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

output

```
YES
```

Note

That's how we can place the pictures in the first test:

And that's how we can do it in the third one.

C. Gerald's Hexagon

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

Gerald got a very curious hexagon for his birthday. The boy found out that all the angles of the hexagon are equal to 120°. Then he measured the length of its sides, and found that each of them is equal to an integer number of centimeters. There the properties of the hexagon ended and Gerald decided to draw on it.

He painted a few lines, parallel to the sides of the hexagon. The lines split the hexagon into regular triangles with sides of 1 centimeter. Now Gerald wonders how many triangles he has got. But there were so many of them that Gerald lost the track of his counting. Help the boy count the triangles.

Input

The first and the single line of the input contains 6 space-separated integers a_1, a_2, a_3, a_4, a_5 and a_6 ($1 \le a_i \le 1000$) — the lengths of the sides of the hexagons in centimeters in the clockwise order. It is guaranteed that the hexagon with the indicated properties and the exactly such sides exists.

Output

Print a single integer — the number of triangles with the sides of one 1 centimeter, into which the hexagon is split.

Examples

input

```
1 1 1 1 1 1
```

output

```
6
```

input

```
1 2 1 2 1 2
```

output

```
13
```

Note

This is what Gerald's hexagon looks like in the first sample:

And that's what it looks like in the second sample:

D. Equivalent Strings

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

Today on a lecture about strings Gerald learned a new definition of string equivalency. Two strings a and b of equal length are called *equivalent* in one of the two cases:

1. They are equal.
2. If we split string a into two halves of the same size a_1 and a_2, and string b into two halves of the same size b_1 and b_2, then one of the following is correct:

    a. a_1 is equivalent to b_1, and a_2 is equivalent to b_2
    b. a_1 is equivalent to b_2, and a_2 is equivalent to b_1

As a home task, the teacher gave two strings to his students and asked to determine if they are equivalent.

Gerald has already completed this home task. Now it's your turn!

Input

The first two lines of the input contain two strings given by the teacher. Each of them has the length from 1 to $200\,000$ and consists of lowercase English letters. The strings have the same length.

Output

Print "YES" (without the quotes), if these two strings are equivalent, and "NO" (without the quotes) otherwise.

Examples

input

```
aaba
abaa
```

output

```
YES
```

input

```
aabb
abab
```

output

```
NO
```

Note

In the first sample you should split the first string into strings "aa" and "ba", the second one — into strings "ab" and "aa". "aa" is equivalent to "aa"; "ab" is equivalent to "ba" as "ab" = "a" + "b", "ba" = "b" + "a".

In the second sample the first string can be splitted into strings "aa" and "bb", that are equivalent only to themselves. That's why string "aabb" is equivalent only to itself and to string "bbaa".

E. Gerald and Giant Chess

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

Giant chess is quite common in Geraldion. We will not delve into the rules of the game, we'll just say that the game takes place on an $h \times w$ field, and it is painted in two colors, but not like in chess. Almost all cells of the field are white and only some of them are black. Currently Gerald is finishing a game of giant chess against his friend Pollard. Gerald has almost won, and the only thing he needs to win is to bring the pawn from the upper left corner of the board, where it is now standing, to the lower right corner. Gerald is so confident of victory that he became interested, in how many ways can he win?

The pawn, which Gerald has got left can go in two ways: one cell down or one cell to the right. In addition, it can not go to the black cells, otherwise the Gerald still loses. There are no other pawns or pieces left on the field, so that, according to the rules of giant chess Gerald moves his pawn until the game is over, and Pollard is just watching this process.

Input

The first line of the input contains three integers: h, w, n — the sides of the board and the number of black cells ($1 \le h, w \le 10^5, 1 \le n \le 2000$).

Next n lines contain the description of black cells. The i-th of these lines contains numbers r_i, c_i ($1 \le r_i \le h, 1 \le c_i \le w$) — the number of the row and column of the i-th cell.

It is guaranteed that the upper left and lower right cell are white and all cells in the description are distinct.

Output

Print a single line — the remainder of the number of ways to move Gerald's pawn from the upper left to the lower right corner modulo $10^9 + 7$.

Examples

input

```
3 4 2
2 2
2 3
```

output

```
2
```

input

```
100 100 3
15 16
16 15
99 88
```

output

```
545732279
```