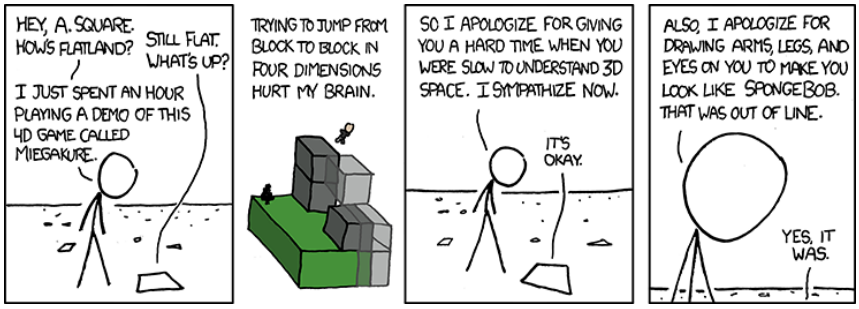
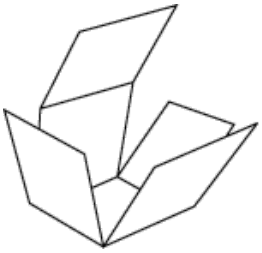
Fold the cube



Description

In order to represent a 3 dimensional cube in 2 dimensions it needs to be unfolded. We will always unfold cubes to look like this:



```
    -----
    | 1 |
-------------
| 2 | 3 | 4 |
-------------
    | 5 |
    -----
    | 6 |
    -----
```

We don't know what orientation the cube was before it was unfolded. Based on the 2D representations of the cubes, we want to know if the cubes could have been the same in 3D after refolding and rotations.

Note that when refolding, we can fold the edges up or down. For example, if you fold the edges up, face 3 would be on the bottom, but if you folded the edges down 3 would be on the top.

Input

The first line is a number N. N testcases follow. Each test case consists of two flatten cubes separated by an empty line. Each face of a cube is represented with a single lowercase letter. A cube can have multiple identical faces.

Output

Output `same` if they are the same, and `different` if they are different.

Examples

Input

```
 b
aba
 b
 b

 b
bab
 b
 a
```

Output

```
same
```

Input

```
 b
adc
 e
 f

 a
adc
 e
 f
```

Output

```
different
```

Input

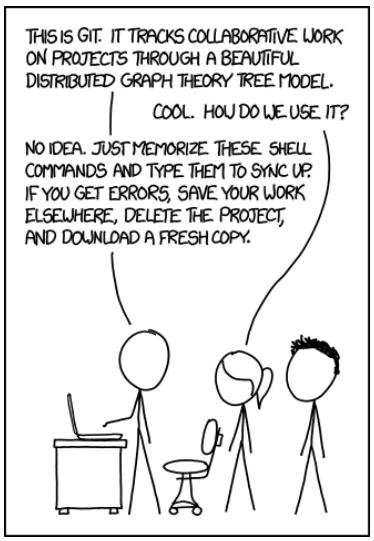
```
 a
bcd
 e
 f

 a
bfd
 e
 c
```

Output

```
same
```

Git



Description

Given an action, choose the correct Git commands to accomplish it, as follows:

send

```
git add -A
git commit -m "Some changes"
git pull
git push
```

receive

```
git add -A
git stash
git pull
git stash pop
```

info

```
git status -s
```

Input

The input is the desired action: send, receive, or info.

Output

Ouput the corresponding Git commands. Don't improvise; give the commands exactly as shown.

Examples

Input

```
receive
```

Output

```
git add -A
git stash
git pull
git stash pop
```

Input

```
info
```

Output

```
git status -s
```

Word to number

Description

You will convert numbers from words to numerals.

The numbers will be nonnegative integers less than one billion. Their word representations will be <number>s according to the following BNF grammar:

```
<ones> ::= "one" | "two" | "three" | "four" | "five" | "six" | "seven" | "eight" | "nine"
<tens-place> ::= "twenty" | "thirty" | "forty" | "fifty" | "sixty" | "seventy" | "eighty"
        | "ninety"
<tens> ::= <ones> | <tens-place> | <tens-place> "-" <ones> | "ten" | "eleven" | "twelve"
        | "thirteen" | "fourteen" | "fifteen" | "sixteen" | "seventeen" | "eighteen" | "nineteen"
<hundreds> ::= <tens> | <ones> hundred | <ones> hundred <tens>
<thousands> ::= <hundreds> | <hundreds> thousand | <hundreds> thousand <hundreds>
<millions> ::= <thousands> | <hundreds> million | <hundreds> million <thousands>
<number> ::= <millions> | "zero"
```

Input

The input is the word representation of the number.

Output

Output the decimal representation of the number.

Examples

Input

```
zero
```

Output

```
0
```

Input

```
seventeen
```

Output

```
17
```

Input

```
fifty-two
```

Output

```
52
```

Input

```
one thousand one
```

Output

```
1001
```

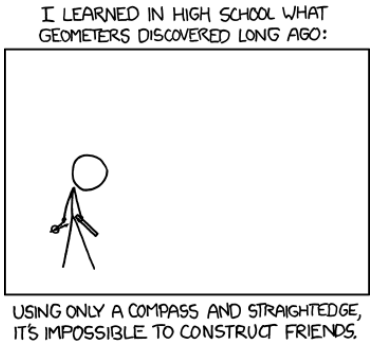
Input

```
nine hundred ninety-nine million nine
 hundred ninty-nine thousand nine hun
dred ninty-nine
```

Output

```
999999999
```

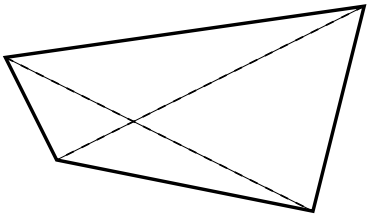
Quad areas



Description

Since you don't have any friends, you get to spend Saturday morning solving a geometry problem.

A convex quadrilateral can be divided into four non-overlapping triangles by connecting opposite vertices, as shown.



You must find the area of the largest of these four triangles.

You may find the "shoelace formula" helpful. It describes the area of a degree-n polygon with vertices $(x_1, y_1), (x_2, y_2), ... (x_n, y_n)$:

$$A = \frac{1}{2}\left|\left(\sum_{i=1}^{n-1} x_i y_{i+1}\right) + x_n y_1 - \left(\sum_{i=1}^{n-1} x_{i+1} y_i\right) - x_1 y_n\right|$$

which, when $n = 3$, is

$$A = \frac{1}{2}\left|x_1 y_2 + x_2 y_3 + x_3 y_1 - x_2 y_1 - x_3 y_2 - x_1 y_3\right|$$

Input

The input is the four vertices, in order. Each vertex is on a line, and each line has the x- and y-coordinates separated by a space. All coordinates are integers from -500 to 500 inclusive.

The quadrilateral is guaranteed to be guarantted convex; i.e. the angle at each vertex is less than 180 degrees.

Output

Output a single number: the area of the largest triangle. This answer must be accurate to within 0.001.

Examples

Input

```
0 0
0 1
1 1
1 0
```

Output

```
0.25
```

Input

```
0 5
10 400
100 500
50 -5
```

Output

```
11588.288
```

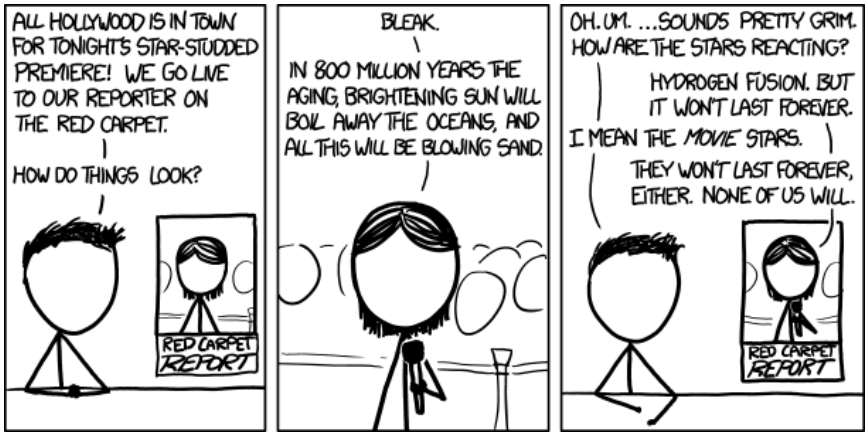
Input

```
-2 0
0 1
2 0
0 -1
```

Output

```
1
```

Movie stars



Description

You're hoping to be a news reporter one day and you're looking for a movie star to interview. Your friend has given you a method for determining if there is a celebrity in the room, and who it is.

He tells you that a person is a celebrity if everyone else in the room knows them, but they don't know anybody else in the room.

Determine whether there is a celebrity in the room, and who it is.

Input

The first line will contain two numbers, N (2 <= N <= 1,000), the number of people in the room, and E (0 <= E <= N*(N-1)), the number of relationships. People are numbers 0 to N-1.

The next E lines will be two space-separated numbers A and B, meaning that person A knows person B.

Output

Print the celibrity's number if there is one. If there is no celebrity, print -1.

Examples

Input

```
4 6
1 2
0 2
0 3
2 3
2 0
1 3
```

Output

```
3
```

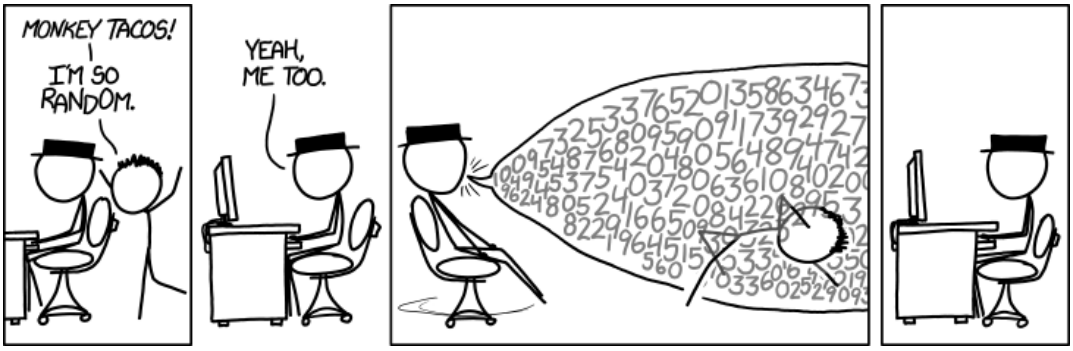
Input

```
4 6
1 2
0 2
0 3
2 3
1 3
3 1
```

Output

```
-1
```

Sum of fractions



Description

Given a list of rational numbers, find the sum in reduced form. A positive fraction is in reduced form if the numerator and denominator are positive integers that have no common divisors greater than 1. (For example, 2/4 in reduced form is 1/2).

Input

The first line is an integer N. (0 < N <= 5)

The next N lines will contain a rational number in the form A/B, where A and B are positive integers less than 500.

Output

Print the sum of the numbers in reduced form.

Examples

Input

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

Output

```
5/6
```

Input

```
2
5/10
2/6
```

Output

```
5/6
```

Input

```
3
3/4
6/5
342/8
```

Output

```
447/10
```

Day of the year

Description

You want to know how many days there have been so far this year.

As a reminder (just in case you forgot the pneumonic), January, March, May, July, August, October, and December have 31 days. April, June, September, and November have 30 days. February has 28 days on non-leap years and 29 days on leap years.

A year is a leap year if:

```
The year is divisible by 4
    UNLESS the year is divisible by 100
        UNLESS the year is divisible by 400.
```

Input

A date in format MM/DD/YYYY.

Output

The day of the year (where 01/01/YYYY is day 1 of the year).

Examples

Input

```
11/07/2015
```

Output

```
311
```

Input

```
04/20/1992
```

Output

```
111
```

Input

```
03/22/1624
```

Output

```
82
```

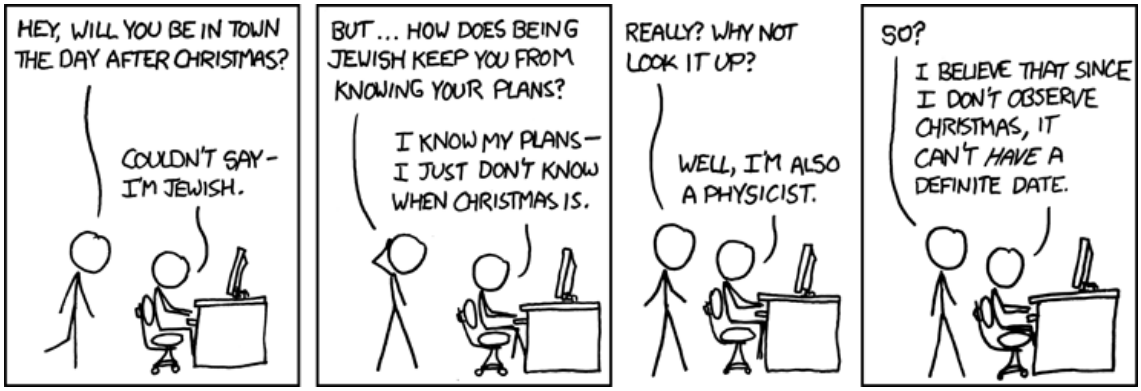
Input

```
01/23/1381
```

Output

```
23
```

Christmas shopping



Description

Lack of planning (and arguing with stubborn physicists) has made you terribly behind on Christmas shopping. The mall is about to close, and you need to buy presents from several stores in the fastest time possible. The stores in the mall are arranged in a loop. You plan to visit each store seqentially in clockwise order, until you are done or the mall closes. It takes one minute to get necessary presents from each store.

Find the best store to start at.

1. Since the mall could close at any time, choose the way to get the most presents within one minute.
2. If there is a tie, choose the way to get the most presents within two minutes.
3. If there is a tie, choose the way to get the most presents within three minutes.
4. And so on.

For example, suppose these are the stores in clockwise order:

- Store 1: two presents
- Store 2: one present
- Store 3: two presents

Starting at store 1 or store 3 gives you the most number of presents in one minute (two presents). Starting at store 3 givs you the most presents in two minutes (four presents).
Therefore, it is best to start at store 3.

Input

There are N (1 <= N <= 200,000) consecutive numerals 1-9, which are the number of presents at each of the N stores in clockwise order, starting with store 1 and ending with store N.
(Remember that the stores are in a loop, so store N is also next to store 1.)

Output

Output the best store to start at, according to the criteria stated earlier. If multiple stores are best, choose the store with the lowest number.

Examples

Input

```
212
```

Output

```
3
```

Input

```
5959
```

Output

```
2
```

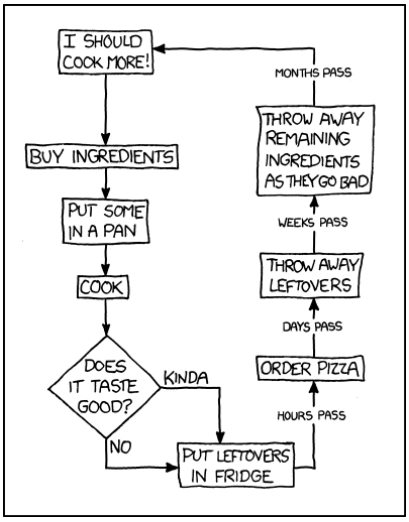
Input

```
11874874874874874874874874874874
```

Output

```
3
```

Pizza slices



Description

Pizza is awesome.

After a programming competition, you and your fellow competitors have pizza for lunch. It's much better than the leftovers in your fridge. You are in charge of slicing and distributing the pizzas, which are of various initial sizes. You may cut each pizza however you like, and as many times as you would like; you could even not cut a pizza at all, and make the whole thing one big "slice". (Yummy.) The only portioning requirement is that each person receive **exactly one** slice of pizza, and they can't share it.

You consider how large of a slice you can guarentee, such that each person gets at least that much. For example, if there are three people, and three pizzas of sizes 1, 2, and 3, you can cut the largest pizza in half, yielding slices of size 1, 2, 1.5, and 1.5. That guarantees each person a slice of size 1.5 or more.

Given that you can cut pizzas however you like, what is largest size slice you can guarantee to everyone?

Input

The first line is two numbers separated by a space, M and N. M is the number of people (0 < M < 1000). N is the number of whole pizzas (0 < N < 1000).
The next N lines are the initial sizes of the pizzas, which are positive integers less than 1000.

Output

Output the maximum size of pizza slice you can guarantee to each person. Your answer must be accurate to within 0.001.

Examples

Input

```
3 3
1
2
3
```

Output

```
1.5
```

Input

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

Output

```
2
```

Input

```
5 5
6
2
3
7
11
```

Output

```
3.667
```

Chinese Checkers

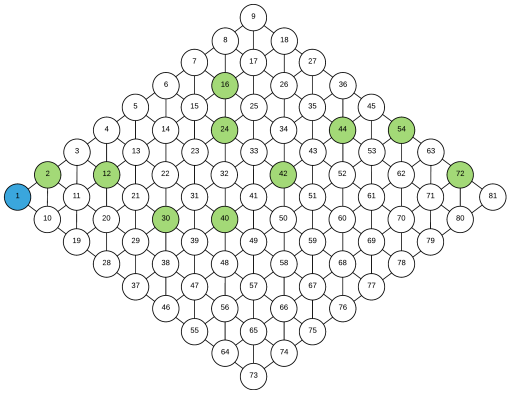
Description

Chinese checkers is a multiplayer game played on hexgram-shaped board with the objective of moving your pieces to the oppostive corner before your opponents do the same.

Each turn, a piece move in one of two ways:

- Move to an adjacent unoccupied cell.
- Jump over an adjacent occupied cell, to an unoccupied cell immediately opposite the occupied one. Consecutive jumps in a single turn are allowed.

The full board is a hexagram (six-pointed star), but we will narrow our focus to the center area, and two opposite corners. There are 81 cells in that region:



Given a starting position, you will get a piece as close to cell 81 as possible in a single turn. In this context, the distance from the goal is considered to be the number of moves it would take to reach it, if there were no occupied cells. For example, cells 54, 62, 70, and 78 are all 3 away from the goal.

Input

There are two lines of input. The first line is a space separated list of occupied cells.

The second line will be a space separated list of origin cells.

Output

For each origin cell, print a line consiting of the closest cell to 81 that can be reached from the origin cell in one turn. There will always be a possible move, and you cannot stay in the same place. If there is a tie in distance, choose the largest number.

Examples

Example 1 is representative of the illustration above. Starting from 1 (in blue), you can navigate to 81 by way of 3, 21, 39, 41, 43, 45, 63, 81. Starting with 7, the furthest you can get is 25, by jumping over 16.

Input

```
2 12 16 24 30 40 42 44 54 72
1 7 17 53 11 41
```

Output

```
81
25
51
62
13
81
```

Input

```
2 12 22 32 42 52 60 70 81
1 11 25 72 80
```

Output

```
79
71
34
80
72
```