Problem A. Arrays Printing

Input file: `arrays.in`
Output file: `arrays.out`

Java language contains arrays as special classes. Since every class in Java derives from `java.lang.Object`, every array does, too. An array of objects can contain other arrays as elements.

Sometimes we should print the given array of objects as a string. To do this, we may use the dedicated method, `java.util.Arrays.deepToString`. Its specification comes below.

> `public static String deepToString(Object[] array)`
>
> Returns a string representation of the "deep contents" of the specified array. If the array contains other arrays as elements, the string representation contains their contents and so on. This method is designed for converting multidimensional arrays to strings.
>
> The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters "`, `" (a comma followed by a space). Elements are converted to strings as by `String.valueOf(Object)`, unless they are themselves arrays.
>
> If an element `e` is an array of a primitive type, it is converted to a string as by invoking the appropriate overloading of `Arrays.toString(e)`. If an element `e` is an array of a reference type, it is converted to a string as by invoking this method recursively.
>
> To avoid infinite recursion, if the specified array contains itself as an element, or contains an indirect reference to itself through one or more levels of arrays, the self-reference is converted to the string "`[...]`". For example, an array containing only a reference to itself would be rendered as "`[[...]]`".
>
> This method returns "`null`" if the specified array is null.

In this problem, we consider only arrays that contain other arrays as elements, or arrays that are empty.

For example, we may take the array

```
Object[] array = new Object[] {
    new Object[0],
    new Object[] {new Object[0], new Object[0]}
};
```

and the invocation of `deepToString` on it will return "`[[], [[], []]]`".

However, two different arrays may give the same string representation. First of all, an array may contain the same object twice or two equal objects. For example, the following two constructions

```
Object[] array = new Object[2];
array[0] = new Object[0];
array[1] = new Object[0];
```

and

```
Object[] array = new Object[2];
array[0] = new Object[0];
array[1] = array[0];
```

will have the same string representation, which would be "`[[], []]`". Note that these constructions are different, because a variable of object type in Java is a reference. As a result, two different arrays may contain the same object as an element, probably more than once.

Given the outcome of the invocation of `deepToString`, find the number of different array configurations that give the specified result. Two array configurations A and B are considered equal if the following statements

- `A`$[a_1][a_2]\ldots[a_k]$ `==` `A`$[b_1][b_2]\ldots[b_m]$
- `B`$[a_1][a_2]\ldots[a_k]$ `==` `B`$[b_1][b_2]\ldots[b_m]$

are equivalent for every possible sequences of a_i and b_j. Here, the relation "`==`" should be treated as "equal by reference", how it is in Java language when the objects are considered.

For the sake of simplicity, you may assume that the given string will be created using the following construction:

```
java.util.deepToString(new Object[a1][a2][a3]...[an][0]);
```

where $a_1, \ldots, a_n$ will be positive integer numbers, $a_1 \cdot a_2 \cdot \ldots \cdot a_n \le 512$, $n \le 100$. Shortly speaking, this will be a string representation of a multidimensional array with constant dimensions.

Input

The input file will consist of a single non-empty string S with length not exceeding 10^5. This string will be an outcome of calling `deepToString` on an array of the described type.

Output

Output the number of arrays with the given string representation modulo 10007.

Sample input and output

arrays.in	arrays.out
`[[], []]`	2
`[[[[]]]]`	1
`[[[], []]]`	2
`[[[]], [[]]]`	3

The arrays in the examples were created by the following statements, respectively.

```
new Object[2][0];
new Object[1][1][1][0];
new Object[1][2][0];
new Object[2][1][0];
```

Problem B. Big Brother

Input file: `brother.in`
Output file: `brother.out`

Alex and Bob are brothers. Alex is the elder brother, and Bob is the younger one. Since the last New Year Day, their parents started giving them pocket money as a fee for household assistance or good marks in school.

The children are in the seventh heaven. They became A-students, run to the shop one ahead of another to buy some bread and clean the room in the sake of making more money. Brothers do not spend their money but store them in two separate piggy-banks: Alex wants to buy a bicycle, and Bob aims to save a million.

During to their looong life together, Alex has always had more benefit than Bob, so he would become angry if at some moment the sum in the piggy-bank of the younger brother becomes greater than in his one. Certainly, he would break all the windows in the nearest house in such case. Bob is above all this — he has a great aim, so such an issue is not worth breaking nerves.

Poor brothers! They don't know that in fact, the sums to be given away for them are determined well ahead. Notably, during this year Alex will be given K euros, and Bob will be given M euros. Each good deed is rewarded by giving 1 euro, and in no case the brothers would be paid simultaneously.

Consider K good deeds of Alex and M those of Bob which will be paid off and assume that every permutation of those deeds is equally probable.

Find the probability for the window in the next house to remain unbroken during the year.

Input

In the first line of the input file there is the number of test cases N.

Each of the next N lines consists of two integers K_i and M_i — the amounts of money to be paid to elder and younger brother, respectively. $0 \le K_i, M_i \le 10^4$.

Output

For each test case, output the answer on a separate line with precision not less than 10^{-6}.

Sample input and output

brother.in	brother.out
3 1 0 0 1 1 1	1 0 0.500

Problem C. Clever House

Input file: clever.in
Output file: clever.out

Little boy Vasya likes high-end technologies. Recently he read about "clever houses", where everything is managed automatically, and decided to make his house "clever". To begin with, he connected all N light-bulbs to his computer, which will help to manage the lighting level and save the energy.

Little hacker Petr, who lives nearby, decided to make a nasty thing to Vasya. He created a computer virus. The main action of this virus is to select a random light-bulb and change its state (to switch off if it was on, and vice versa). This action is repeated for K times.

At the beginning there were M light bulbs turned on. Now Petr wants to know how efficient his attack will be. More precisely, he wants to know how much light-bulbs in average will be on after the virus is activated (the mathematical expectation of this value). Help him in this complex task!

Input

In the first and only line there are three integer numbers: N $(0 < N \leq 10^9)$, M $(0 \leq M \leq N)$, K $(0 \leq K \leq 1000)$.

Output

Output the answer as an irreducible fraction p/q.

Sample input and output

clever.in	clever.out
1 1 9	0/1
5 4 3	353/125

Problem D. Definite Integral

Input file: `definite.in`
Output file: `definite.out`

Your task is to evaluate the following definite integral:

$$\int_{-\infty}^{+\infty} \frac{dx}{P(x)}$$

where $P(x) = a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$. $P(x)$ has no real roots and $GCD(P(x), P'(x)) = const$.

Input

The input file contains five integer numbers: a_0, a_1, a_2, a_3 and a_4 separated by whitespace. Each of these numbers does not exceed 10^6 by absolute value, $a_4 \neq 0$.

Output

Output the value of the integral.

Assume that the exact value is A and your answer is B. Your answer will be considered correct if and only if at least one of the following statements is true:

- $A - 10^{-9} \leq B \leq A + 10^{-9}$
- $A \cdot (1 - 10^{-9}) \leq B \leq A \cdot (1 + 10^{-9})$

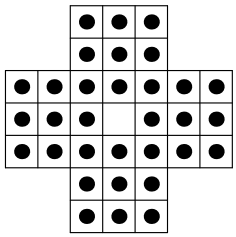
Sample input and output

definite.in	definite.out
16 0 0 0 1	0.2776801836

Problem E. Endspiel

Input file: `endspiel.in`
Output file: `endspiel.out`

There is a game called "Yoga". On the game board there are 32 checkers, standing as shown on the picture. Each turn a checker jumps over another one and lands on a free cell — almost like in the checker game, but vertically or horizontally, not diagonally. The checker which was jumped over is removed from the board.



We will look at endspiel, the last part of the game. Imagine that there is only one checker left. Given its location, find a possible sequence of turns that leads to this endspiel.

Input

Let us introduce a coordinate system similar to the one that is used in the game of chess. The columns are numbered by Latin letters from `A` to `G`, the rows are numbered from `1` to `7`. For example, a cell with coordinates `D4` is the central cell.

The first line of the input file contains the coordinates of the last checker in the notation described above.

Output

If it is possible to reach the specified endspiel, output the sequence of turns leading to it. Each turn should be printed in the following format: `<start_cell>-<finish_cell>`, where `<start_cell>` is the coordinates of a cell where the moving checker is located before the turn, and `<finish_cell>` is the coordinates of its destination cell. There will always be 31 turns.

If it is impossible to find the necessary sequence, output the word "`Impossible`".

Sample input and output

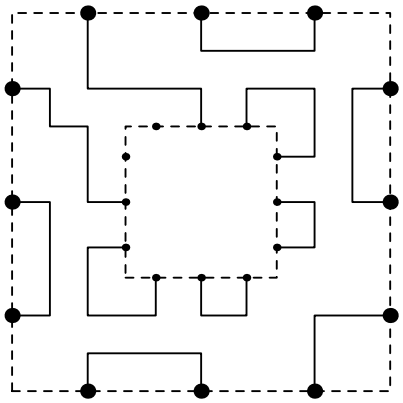
endspiel.in	endspiel.out
D3	Impossible
D4	B4-D4 C6-C4 A5-C5 ... C5-C3 B3-D3 D2-D4

Problem F. Fractal Labyrinth

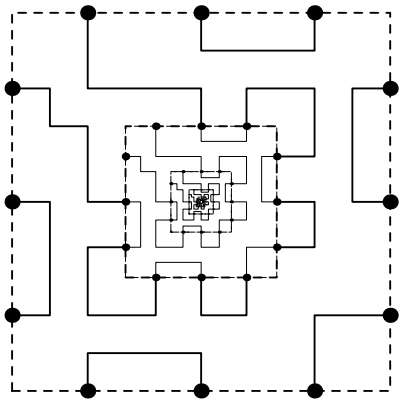
Input file: `fractal.in`
Output file: `fractal.out`

Roman adores all kinds of labyrinths. He solves them since his early childhood. Yesterday he found a new, extremely amazing sort of labyrinth. He called it a "fractal labyrinth".

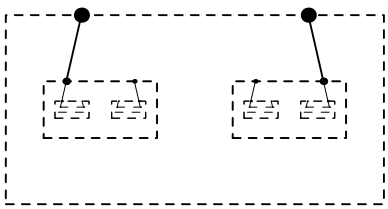
Imagine a house with N doors. Inside it, there are K houses, each of them an entire copy of the "outer" one. Some doors are connected by roads. If you draw these roads, you get something like this:



Remembering that each "inner" house is a copy of the "outer house", you get the following picture as a result:



The following picture is an example of a house with 2 inner houses:



For outer house, some door is defined as "input" and some door as "output", so we finally come up to a labyrinth. Assuming that length of each road is 1, find the length of the shortest path in such labyrinth.

Input

In the first line of input file there are numbers N ($2 \le N \le 20$) and K ($0 \le K \le 5$). The next line contains M, the number of the roads. In the following M lines there are descriptions of the roads, one per line.

Each road description is formatted in the following way:

`<house-number>.<door-number> - <house-number>.<door-number>`

where left and right part of description specify the connected doors (a door is described by its number and number of house it belongs to). Each road is bidirectional. The "outer" house has number 0, "inner" houses have numbers from 1 to K. Door numbers start with 0. No two roads connect the same pair of doors. No road connects a door to itself.

In the last line there are numbers of input and output door D_i and D_o. These numbers may conincide.

Output

If there exists a path from input to output, print the minimal length of such path. Otherwise, print "`no solution`".

Sample input and output

fractal.in	fractal.out
12 1 11 0.0 - 1.1 0.1 - 0.2 1.2 - 1.3 0.3 - 0.4 1.4 - 1.5 0.5 - 0.6 1.6 - 1.7 0.7 - 0.8 1.8 - 1.9 0.9 - 0.10 1.10 - 0.11 0 11	11
8 0 8 0.0 - 0.2 0.1 - 0.3 0.2 - 0.4 0.3 - 0.5 0.4 - 0.6 0.5 - 0.7 0.6 - 0.0 0.7 - 0.1 2 5	no solution

Problem G. Go out of here!

Input file: `goout.in`
Output file: `goout.out`

Two scouts were sent on a top-secret mission. Unfortunately, they were captured and put in prison.

The prison is a rectangular one-storey building having walls and corridors (doors of prison cells are considered to be walls). The plan of the building is a checkered rectangle with square size equal to 2 metres. Some of square sides are marked in bold, and such sides correspond to the walls. Surely, all the outer square sides are walls — if not, what a sort of prison would it be?

At last, scouts gnawed through doors of their cells and left them. They had to get out of the prison as soon as they could. However, there was an aggravating circumstance: they had been made blind (luckily, only temporarily) by special medicine. Still, both of them had transmitters embedded under their skin, so they could communicate with a base to a limited extent.

The problem was that the scouts didn't know their coordinates, even relative to the prison. Furthermore, since they were kept in different cells, they were unable neither to communicate, nor even to see each other.

So the headquaters ordered to proceed as follows: scouts in turn try to move in either of the four directions (North, South, East or West — what sort of a scout you are if you can't find north with eyes closed!) and report the direction and whether the move was successful or not. For simplicity, the move distance was selected to be 2 metres, so the headquaters can assume that a scout occupy a square on the prison's plan.

The headquaters became aware of the size of the prison and got informed that its walls were parallel to the North-to-South and East-to-West lines. Soon enough, the scouts were located by their reports and were helped to get out.

You are now assigned a task to determine how effective the rescue operation was. You are given the records of reports of the scouts. Now, determine the minimal number of steps made by scouts that were necessary to determine precisely the coordinates of the scouts.

Input

The first line of the input file contains the prison width W and height H, and the number of scouts' reports K ($2 \le W \cdot H \le 150, 1 \le K \le 10^5$). W is the number of squares in the direction from East to West, and H — from North to South.

The reports of the scouts are located in each of the following K lines. Each report consists of two symbols: (`N`, `S`, `E`, `W`) — the direction and (`+`, `-`) — whether the move was successful or not. The odd reports are of the first scout, the even are of the second one.

If a scout makes a successful move, he moves to the destination square. Otherwise, he remains in the same square where he started from.

Output

If it is possible to determine the number of report **after** which the coordinates of both scouts become known, output "`The scouts are safe at step number`" and the one-based report number.

If it is impossible to determine the coordinates, output "`There is not enough data`".

If there are mistakes (in the reports, not in the input format), output "`A mistake has been made at step number`" and the number of first erroneous (that is, incompatible with any of the previous) report. Note that you should output this even if there is a mistake after the moment the coordinates are determined.

Sample input and output

goout.in	goout.out
2 1 4 E+ W- N- S-	The scouts are safe at step number 2
2 1 4 N- W- N- S-	There is not enough data
2 1 4 N- W- N- S+	A mistake has been made at step number 4

Problem H. Hankel Matrix

Input file: `hankel.in`
Output file: `hankel.out`

A *Hankel matrix* is a matrix of the following form:

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_n \\ \alpha_2 & \alpha_3 & \alpha_4 & & \vdots \\ \alpha_3 & \alpha_4 & \alpha_5 & & \vdots \\ \vdots & & & \ddots & \vdots \\ \alpha_n & \dots & \dots & \dots & \alpha_{2n-1} \end{pmatrix}$$

That is, $A_{i,j} = F(i + j)$.

Find an integer Hankel matrix of the given size with determinant equal to one. Moreover, all its square submatrices containing upper left cell must also have determinant equal to one.

Input

The input file contains the size of the matrix N ($1 \le N \le 200$).

Output

Output $(2N - 1)$ integer numbers — the numbers α_1, α_2, ..., α_{2N-1} from which the matrix is built, one per line.

Sample input and output

hankel.in	hankel.out
1	1
2	1 6 37
3	1 3 10 7 630

Problem I. Interfering Segment

Input file: `interfering.in`
Output file: `interfering.out`

Consider a simple polygon P (that is, polygon without self-intersections) and a segment S lying strictly inside it.

A *triangulation* of a polygon P is its partition into non-overlapping triangles whose union is P. In this problem, we put some restrictions on triangulations: all vertices of a triangle must coincide with some vertices of P and no vertex of P must lie on a boundary of a triangle (except for triangle's vertices). We call a segment *interfering* with a triangulation if it intersects (or touches) a boundary of some triangle of the triangulation.

Your task is, given the polygon P and segment S, to determine whether there exists a triangulation that S does not interfere with.

Since it is well-known that every simple polygon can be triangulated, you have only to output the triangle that belongs to the triangulation you found and contains S strictly inside.

Input

In the first line there is N ($3 \le N \le 800$) — the number of vertices in P.

The following N lines contain pairs of integers (X_i, Y_i) — the coordinates of vertices of P. All points are distinct, and no three consecutive points lie on the same line.

The last line contain four integers X_s, Y_s, X_f, Y_f — the coordinates of endpoints of S.

All coordinates do not exceed 10^4 by absolute value. The segment S is guaranteed to lie strictly inside the polygon P. S is also guaranteed to have non-zero length.

Output

If the solution does exist, output the one-based indices of vertices of triangle that belongs to some triangulation and contains S strictly inside. The indices must be output on a single line and separated by single spaces.

If the solution does not exist, output the word "`Impossible`" on a single line.

Sample input and output

interfering.in	interfering.out
3 0 0 0 3 4 3 1 2 2 2	1 2 3
4 0 0 2 0 2 3 0 3 1 1 1 2	Impossible

Problem J. Join

Input file: `join.in`
Output file: `join.out`

Businessman Petya recently bought a new house. This house has one floor, and has $n \times m$ square rooms, placed in rectangular lattice. Now he wants to join all rooms except pantries with doors in such a way that there will be exactly one way between any pair of them. He can make doors only between neighbouring rooms (i.e. rooms having a common wall).

Now he wants to count the number of different ways he can do it.

Input

First line contains two integers n and m ($1 \le n, m \le 9$) — the number of lines and columns in the lattice. Next n lines contains exactly m characters representing house map, where "`.`" means room and "`*`" means pantry.

Output

Output the number of ways to join rooms modulo 10^9.

Sample input and output

join.in	join.out
2 2	4
2 2 *. .*	0