A. Plate Game

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

You've got a rectangular table with length a and width b and the infinite number of plates of radius r. Two players play the following game: they take turns to put the plates on the table so that the plates don't lie on each other (but they can touch each other), and so that any point on any plate is located within the table's border. During the game one cannot move the plates that already lie on the table. The player who cannot make another move loses. Determine which player wins, the one who moves first or the one who moves second, provided that both players play optimally well.

Input

A single line contains three space-separated integers a, b, r ($1 \le a, b, r \le 100$) — the table sides and the plates' radius, correspondingly.

Output

If wins the player who moves first, print "`First`" (without the quotes). Otherwise print "`Second`" (without the quotes).

Examples

input

```
5 5 2
```

output

```
First
```

input

```
6 7 4
```

output

```
Second
```

Note

In the first sample the table has place for only one plate. The first player puts a plate on the table, the second player can't do that and loses.

In the second sample the table is so small that it doesn't have enough place even for one plate. So the first player loses without making a single move.

B. Limit

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

You are given two polynomials:

- $P(x) = a_0 \cdot x^n + a_1 \cdot x^{n-1} + \ldots + a_{n-1} \cdot x + a_n$ and
- $Q(x) = b_0 \cdot x^m + b_1 \cdot x^{m-1} + \ldots + b_{m-1} \cdot x + b_m$.

Calculate limit $\lim_{x \to +\infty} \frac{P(x)}{Q(x)}$.

Input

The first line contains two space-separated integers n and m ($0 \le n, m \le 100$) — degrees of polynomials $P(x)$ and $Q(x)$ correspondingly.

The second line contains $n + 1$ space-separated integers — the factors of polynomial $P(x)$: a_0, a_1, ..., a_{n-1}, a_n $(-100 \le a_i \le 100, a_0 \ne 0)$.

The third line contains $m + 1$ space-separated integers — the factors of polynomial $Q(x)$: b_0, b_1, ..., b_{m-1}, b_m $(-100 \le b_i \le 100, b_0 \ne 0)$.

Output

If the limit equals $+\infty$, print "`Infinity`" (without quotes). If the limit equals $-\infty$, print "`-Infinity`" (without the quotes).

If the value of the limit equals zero, print "`0/1`" (without the quotes).

Otherwise, print an irreducible fraction — the value of limit $\lim_{x \to +\infty} \frac{P(x)}{Q(x)}$, in the format "`p/q`" (without the quotes), where p is the — numerator, q $(q > 0)$ is the denominator of the fraction.

Examples

input

```
2 1
1 1 1
2 5
```

output

```
Infinity
```

input

```
1 0
-1 3
2
```

output

```
-Infinity
```

input

```
0 1
1
1 0
```

output

```
0/1
```

input

```
2 2
2 1 6
4 5 -7
```

output

```
1/2
```

input

```
1 1
9 0
-5 2
```

output

```
-9/5
```

Note

Let's consider all samples:

1. $\lim\limits_{x \to +\infty} \frac{x^2+x+1}{2x+5} = +\infty$
2. $\lim\limits_{x \to +\infty} \frac{-x+3}{2} = -\infty$
3. $\lim\limits_{x \to +\infty} \frac{1}{x} = 0$
4. $\lim\limits_{x \to +\infty} \frac{2x^2+x+6}{4x^2+5x-7} = \frac{1}{2}$
5. $\lim\limits_{x \to +\infty} \frac{9x}{-5x+2} = -\frac{9}{5}$

You can learn more about the definition and properties of limits if you follow the link: `http://en.wikipedia.org/wiki/Limit_of_a_function`

C. Lexicographically Maximum Subsequence

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

You've got string s, consisting of only lowercase English letters. Find its lexicographically maximum subsequence.

We'll call a non-empty string $s[p_1p_2... p_k] = s_{p_1}s_{p_2}... s_{p_k}(1 \le p_1 < p_2 < ... < p_k \le |s|)$ a *subsequence* of string $s = s_1s_2... s_{|s|}$.

String $x = x_1x_2... x_{|x|}$ is *lexicographically larger* than string $y = y_1y_2... y_{|y|}$, if either $|x| > |y|$ and $x_1 = y_1, x_2 = y_2, ... , x_{|y|} = y_{|y|}$, or exists such number r $(r < |x|, r < |y|)$, that $x_1 = y_1, x_2 = y_2, ... , x_r = y_r$ and $x_{r+1} > y_{r+1}$. Characters in lines are compared like their ASCII codes.

Input

The single line contains a non-empty string s, consisting only of lowercase English letters. The string's length doesn't exceed 10^5.

Output

Print the lexicographically maximum subsequence of string s.

Examples

input
```
ababba
```
output
```
bbba
```

input
```
abbcbccacbbcbaaba
```
output
```
cccccbba
```

Note

Let's look at samples and see what the sought subsequences look like (they are marked with uppercase bold letters).

The first sample: a**B**a**BBA**

The second sample: abb**C**b**CC**a**C**bb**CB**aa**BA**

D. Infinite Maze

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

We've got a rectangular $n \times m$-cell maze. Each cell is either passable, or is a wall (impassable). A little boy found the maze and cyclically tiled a plane with it so that the plane became an infinite maze. Now on this plane cell (x, y) is a wall if and only if cell $(x \bmod n, y \bmod m)$ is a wall.

In this problem $a \bmod b$ is a remainder of dividing number a by number b.

The little boy stood at some cell on the plane and he wondered whether he can walk infinitely far away from his starting position. From cell (x, y) he can go to one of the following cells: $(x, y - 1)$, $(x, y + 1)$, $(x - 1, y)$ and $(x + 1, y)$, provided that the cell he goes to is not a wall.

Input

The first line contains two space-separated integers n and m ($1 \le n, m \le 1500$) — the height and the width of the maze that the boy used to cyclically tile the plane.

Each of the next n lines contains m characters — the description of the labyrinth. Each character is either a "#", that marks a wall, a ".", that marks a passable cell, or an "S", that marks the little boy's starting point.

The starting point is a passable cell. It is guaranteed that character "S" occurs exactly once in the input.

Output

Print "Yes" (without the quotes), if the little boy can walk infinitely far from the starting point. Otherwise, print "No" (without the quotes).

Examples

input

```
5 4
##.#
##S#
#..#
#.##
#..#
```

output

```
Yes
```

input

```
5 4
##.#
##S#
#..#
..#.
#.##
```

output

```
No
```

Note

In the first sample the little boy can go up for infinitely long as there is a "clear path" that goes vertically. He just needs to repeat the following steps infinitely: up, up, left, up, up, right, up.

In the second sample the vertical path is blocked. The path to the left doesn't work, too — the next "copy" of the maze traps the boy.

E. Paint Tree

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

You are given a tree with n vertexes and n points on a plane, no three points lie on one straight line.

Your task is to paint the given tree on a plane, using the given points as vertexes.

That is, you should correspond each vertex of the tree to exactly one point and each point should correspond to a vertex. If two vertexes of the tree are connected by an edge, then the corresponding points should have a segment painted between them. The segments that correspond to non-adjacent edges, should not have common points. The segments that correspond to adjacent edges should have exactly one common point.

Input

The first line contains an integer n ($1 \le n \le 1500$) — the number of vertexes on a tree (as well as the number of chosen points on the plane).

Each of the next $n - 1$ lines contains two space-separated integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the numbers of tree vertexes connected by the i-th edge.

Each of the next n lines contain two space-separated integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the i-th point on the plane. No three points lie on one straight line.

It is guaranteed that under given constraints problem has a solution.

Output

Print n distinct space-separated integers from 1 to n: the i-th number must equal the number of the vertex to place at the i-th point (the points are numbered in the order, in which they are listed in the input).

If there are several solutions, print any of them.

Examples

input

```
3
1 3
2 3
0 0
1 1
2 0
```

output

```
1 3 2
```

input

```
4
1 2
2 3
1 4
-1 -2
3 5
-3 3
2 0
```

output

```
4 2 1 3
```

Note

The possible solutions for the sample are given below.