Problem A. Aviation Reform

Input file:	`avia.in`
Output file:	`avia.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

There are n cities in Perverland, and there are m flights between these cities. People of Perverland fly so often that there are at least $\min(n-1, n/2+8)$ flights from each city.

All these flights are operated by Perver Avia company owned by the country government. Recently the liberal government has got to power in Perverland, so they decided to privatize the company.

To stimulate competition it was decided that the company would be split to two companies, Per Avia and Ver Avia. All existing flights will be divided between these companies. In order to ensure that people can diversify their flight, the following condition must be satisfied for each two cities u and v:

- either there is a direct flight between u and v;
- or there is a city w such that there are direct flight between u and w and between v and w, respectively, that belong to different airlines.

Now the ministry of transport wonder how they can perform the privatization. Help them to find that out.

Input

The input file contains n ($4 \le n \le 1000$). The following n lines contain a sequence of n characters each — the j-th character of the i-th line is '`+`' if there is a flight between the i-th city and the j-th city, and '`-`' if there is none. The i-th character of the i-th line is '`-`'. If the j-th character of the i-th line is '`+`' then the i-th character of the j-th line is also '`+`'.

Output

Output the same matrix as in input file, but for each '`+`' output either '`1`' if the corresponding flight must be privatized to Per Avia or '`2`' if it must be privatized to Ver Avia.

If it is impossible to perform the privatization, output "`Impossible`" instead.

Examples

avia.in	avia.out
4 -+++ +-++ ++-+ +++-	-111 1-11 11-1 111-
20 --++++++++++++++++++ --++++++++++++++++++ ++--++++++++++++++++ ++--++++++++++++++++ ++++--++++++++++++++ ++++--++++++++++++++ ++++++--++++++++++++ ++++++--++++++++++++ ++++++++--++++++++++ ++++++++--++++++++++ ++++++++++--++++++++ ++++++++++--++++++++ ++++++++++++--++++++ ++++++++++++--++++++ ++++++++++++++--++++ ++++++++++++++--++++ ++++++++++++++++--++ ++++++++++++++++--++ ++++++++++++++++++-- ++++++++++++++++++--	--11111111111111112 --11111111111111121 11--111111111111211 11--111111111112111 1111--11111111121111 1111--11111111211111 111111--111112111111 111111--111121111111 11111111--1211111111 11111111--2111111111 1111111112--11111111 1111111121--11111111 111111121111--111111 111111211111--111111 11111211111111--1111 11112111111111--1111 1112111111111111--11 1121111111111111--11 12111111111111111-- 21111111111111111--

Problem B. Busy Beavers

Input file:	`beavers.in`
Output file:	`beavers.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Beavers have a natural trait of building dams in rivers and streams. Beavers gnaw the bark off the trees that are later used to build the dam.

Let us consider the process of preparing for building a dam in details. The beaver is standings in front of the line of trees. We will consider the line to be infinite in both directions. Each tree in the line can be *nice* or *ugly*. Initially all trees are nice. However after the beaver gnaws some bark from the tree it can become ugly. Similarly, ugly tree can also be turned back to nice by the beaver.

The beaver can be in one of 5 moods: *angry*, *busy*, *creative*, *despaired* or *exhausted*. Initially the beaver is angry.

When the beaver is standing in front of the tree, depending on its mood and the state of the tree, it does the following:

- gnaws some bark, either turning the tree from nice to ugly and vice versa, or not;
- changes his mood to any of the five possible moods;
- moves either to the tree on the left, or to the tree on the right.

Also in some combination of the beaver's mood and the tree state the beaver can become *happy*, decide that the trees are ready to build the dam and stop gnawing the trees. However, it can happen that the beaver never gets happy.

Given the rules the beaver acts by, find out whether the beaver will get happy and proceed to building the dam, or it will gnaw the trees forever.

Input

The input file contains 10 tokens separated by spaces. The tokens describe the beaver's behavior, the first token describes the angry beaver's behavior on a nice tree, the second token describes the angry beaver's behavior on an ugly tree, the third token describes the busy beaver's behavior on a nice tree, and so on, the tenth token describes the exhausted beaver's behavior on an ugly tree.

Each token consists of three character. The first character is '`1`' if the beaver leaves the current tree ugly, or '`0`' if the beaver leaves the current tree nice. The second character is '`R`' if the beaver moves to the tree on the right, or '`L`' if the beaver moves to the tree on the left. Finally the third character is either a character from '`A`' to '`E`' — the new mood of the beaver, or '`H`' if the beaver gets happy.

Output

Output "`happy beaver`" if the beaver eventually gets happy, or "`unhappy beaver`" if the beaver never gets happy.

Examples

beavers.in	beavers.out
1RB 1LC 1RC 1RB 1RD 0LE 1LA 1LD 1RH 0LA	happy beaver
0RA 0RA 0RA 0RA 0RA 0RA 0RA 0RA 0RA 0RA	unhappy beaver

Problem C. Center of the Universe

Input file:	`center.in`
Output file:	`center.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Recently the scientists of Flatland have found the ruins of a city built by some ancient alien civilization. After deciphering some writings found in the ruins, the scientists were struck by the fact — the civilization originated from the center of the universe. Now the space expedition of the center of the universe must be organized!

The problem is — where is the center of the universe? Ancient documents claim that the center can be found using the following method. Consider $2n$ brightest stars of the universe. Organize them into n pairs so that there were no other star on a line drawn through any two paired stars. Then draw a line through each chosen pair of stars. If the pairs are correct, all these lines intersect at one point. This point is the center of the universe.

Scientists have found $2n$ brightest stars, but they cannot organize them into n pairs in the described way. Help them to find the center of the universe, or find out that the ancient texts were deciphered incorrectly.

Input

The first line of the input file contains n ($2 \le n \le 70$). The following $2n$ lines contain coordinates of $2n$ brightest stars of the universe. The universe where Flatland resides is 2-dimensional, all coordinates do not exceed 10^3 by their absolute values, all stars are located at different points of the universe.

Output

If there is exactly one way to organize stars into n pairs so that the conditions of the problem were satisfied, output "`Center of the universe found at (`x`, `y`)`". Print x and y with absolute or relative error of at most 10^{-9}.

If there are several ways to organize the stars into pairs, output "`Ambiguity`". If there is no way to do it, output "`Impossible`".

Examples

center.in	center.out
2 0 0 1 1 1 0 0 1	Center of the universe found at (0.5, 0.5)
2 1 1 10 10 10 0 0 10	Ambiguity
2 0 0 1 1 2 2 3 3	Impossible

Problem D. Cyclic Index

Input file:	`cyclic.in`
Output file:	`cyclic.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A *context free grammar* consists of Σ — the set of characters, N — the set of variables, $S \in N$ — the starting variable, and a finite set $P \subset N \times (N \cup \Sigma)^*$ — the set of productions. Here A^* denotes the set of all words composed of characters from A, including the empty word ε. The production $\langle A, \alpha \rangle \in P$ is usually denoted as $A \to \alpha$. In this problem $\Sigma = \{a, b, \ldots, z\}$ and variables are denoted by uppercase letters of the English alphabet.

Let β and γ be words over $N \cup \Sigma$. If $A \to \alpha \in P$, $\beta = \beta_1 A \beta_2$, $\gamma = \beta_1 \alpha \beta_2$ we say that β transforms to γ and write $\beta \Rightarrow \gamma$. In other words, if $A \to \alpha \in P$ it is allowed to take β and replace any occurrence of A with the word α. We will say $\beta \Rightarrow^* \gamma$ if γ can be obtained from β with zero or more transformations.

For example, let $P = \{S \to aSbS, S \to \varepsilon\}$. In this case $S \Rightarrow^* aabbab$ since

$$S \Rightarrow aSbS \Rightarrow aaSbSbS \Rightarrow aabSbS \Rightarrow aabbS \Rightarrow aabbaSbS \Rightarrow aabbabS \Rightarrow aabbab.$$

A *cyclic index* of context free grammar Γ denoted by $cyc(\Gamma)$ is maximal k such that $S \Rightarrow^* \alpha\beta^k\gamma$ where $\alpha, \beta, \gamma \in \Sigma^*$ and $\beta \neq \varepsilon$. If there is no maximal k we say that $cyc(\Gamma) = +\infty$.

You are given a context free grammar Γ. Find out whether its cyclic index is infinite.

Input

The first line of the input file contains two integer numbers: n — the number of productions in the given grammar ($1 \le n \le 1000$). The following n lines contain one production each. The length of the right part of each production doesn't exceed 10. The starting variable of the grammar is `S`.

Output

Output either "`finite`" or "`infinite`".

Examples

cyclic.in	cyclic.out
2 S->aSbS S->	infinite
4 S->aA S->bbB A->aaB B->abb	finite

Problem E. New Hierarchy

Input file:	`hierarchy.in`
Output file:	`hierarchy.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are n workers in "Mocrosoft" company. Each worker of the company has his own *competence index* which is represented by an integer number from 1 to n. Competence indexes of all workers are different. The higher is the competence number, the more important the worker is. The owner of the company Bill Hates has competence index n.

Due to financial crisis, Bill Hates has decided to reorganize the company to convert its structure to a tree-like hierarchy. After the conversion each worker except Bill himself would have exactly one direct chief. The chief of each worker must have a higher competence index than the worker himself. To ensure the stability of the system no worker must have more direct subordinates than his chief has.

It turned out that there can be several ways to organize the hierarchy in such way. First Bill would like to find the number of ways to do it. Help him.

Input

The input file contains one integer number n ($1 \le n \le 150$).

Output

Output one integer number — the number of ways to organize the hierarchy.

Examples

hierarchy.in	hierarchy.out
4	5

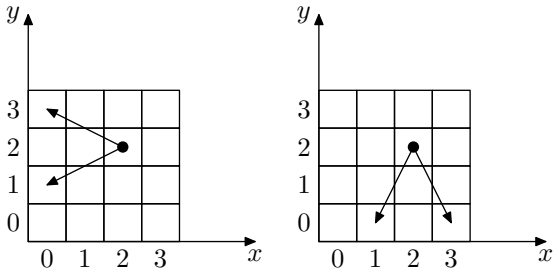
Problem F. Knights

Input file:	`knights.in`
Output file:	`knights.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Andrew and Peter used to play chess every day. However the chessboard is finite, so the number of positions on it is finite. Therefore the friends soon found the game of chess quite boring. So they developed the new version of the game where the board is infinite, so the number of possible positions is infinite.

The players position n knights on the infinite chessboard with cells indexed by two integers (x, y) where $0 \le x$, $0 \le y$. Initially the i-th knight is positioned at cell (x_i, y_i). The players make moves in turn, Andrew moves first.

On his move the player must move each of the knights. Andrew is allowed to move the knight from the cell (x, y) to either cell $(x-2, y-1)$ or to cell $(x-2, y+1)$ (if one of them doesn't exist, Andrew cannot make that move). Similarly Peter can move the knight from the cell (x, y) to one of the cells $(x-1, y-2)$ and $(x+1, y-2)$. If the player to move cannot move at least one knight, he loses. It is allowed to place several knights to the same cell.



Given initial positions of the knights, find out who wins the game if both players play optimally.

Input

The input file contains several test cases. Each test case starts with n — the number of knights ($1 \le n \le 1000$). The following n lines contain two integer numbers each: x_i and y_i ($0 \le x_i, y_i \le 10^9$). Test case with $n = 0$ terminates the input file, it must not be processed.

Output

For each test case print either "`Andrew wins the game`" or "`Peter wins the game`".

Examples

knights.in	knights.out
2 2 2 2 3 1 4 4 0	Andrew wins the game Peter wins the game

Problem G. Nim for Three

Input file:	`nimfor3.in`
Output file:	`nimfor3.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The game of nim is played as follows. Initially there are several heaps of stones in front of players. The players move in turn. The player to move can choose any heap of stones and take one or more stones from it (she can remove the whole heap if she wishes). The player who takes the last stone wins the game. For the sake of completeness we say that if there are initially no stones on the table, the last player to move wins.

Unlike games for two players that are usually antagonistic — forcing the other player to lose allows you to win, games for three or more players can be more complicated. For example, consider the game of nim with two heaps of 1 and 2 stones, respectively. The player to move first cannot win. However, he can choose whether the second player will win (by taking all stones from one of the heaps), or the third one will (by taking one stone from the heap with 2 stones).

Generally, a position in a game can be qualified as N-position (the first player can win regardless of others' moves, the example of such position is "1"), O-position (the second player can win regardless of others' moves, the example of such position is "1 1"), P-position (the third player can win regardless of others' moves, the example of such position is "1 1 1") or Q-position if there is no player that can win regardless of others' moves (the example of such position is "1 2").

For combinatorial games of nim type the notion of the sum of games is often used. The sum of games A and B is the game where the player to move first chooses one of the games: A or B and then makes a move in it. If there is no valid move in any of the games, the player who has made the last move wins. For example, the nim game with two heaps of a and b stones, can be interpreted as a sum of two one-heap nim games (with a and b stones).

In games for two players there can be only N and P-positions. It is known that the sum of two P-games is P-game again, the sum of N and P-games is N-game, and the sum of N and N-games can be both N and P-game depending on the exact games being added.

Your task is to qualify nim games for three players regarding the type of the sum of two games given types of the corresponding games. You are given C_1, C_2 and C_3, each from the set $\{N, O, P, Q\}$. You must detect whether the sum of C_1-position nim game and C_2-position nim game can be C_3-position nim game. You can assume that if it is possible, there are such games involving at most 20 stones all together.

Input

The input file contains three characters: C_1, C_2 and C_3, each from the set $\{N, O, P, Q\}$.

Output

If the sum of C_1-position nim game and C_2-position nim game can be C_3-position nim game, output "`YES`" at the first line of the output file. The following two lines must describe the games. Each game must be described by the sizes of the heaps separated by spaces. If one of the games has 0 stones on the table initially, print one 0 to describe such game.

If the sum of C_1-position nim game and C_2-position nim game cannot be C_3-position nim game, output "`NO`" at the first line of the output file.

Examples

nimfor3.in	nimfor3.out
PPP	YES 0 0
NNN	YES 1 1 2 1
NNP	NO

Problem H. Expected Number of Points

Input file:	`points.in`
Output file:	`points.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are n points located on the plane. No four points are on the same circumference, and no three points are on the same straight line.

Three distinct points are selected at random and a circumference is drawn through the selected points.

Find the expected number of points located strictly inside this circumference.

Input

The first line of the input file contains n — the number of points ($3 \le n \le 1500$). The following n lines contain two integer numbers each — the coordinates of the points. The coordinates do not exceed 10^6 by their absolute values.

Output

Output one number — the expected number of points inside the circumference drawn through three of the given points selected at random. Your answer must be accurate up to 10^{-9}.

Examples

points.in	points.out
4 1 1 10 10 10 0 0 10	0.5
4 7 7 10 10 10 0 0 10	0.25

Problem I. Railroad Simulator

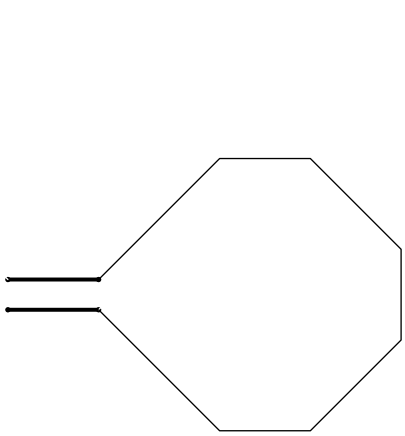
Input file:	`railroad.in`
Output file:	`railroad.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

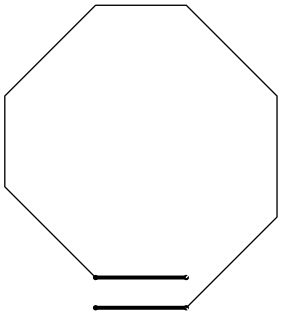
Peter is playing a railroad simulator game on his computer. Initially he would like to build two stations connected by a railroad track and run a train with n cars along it.

The game playing area is an infinite plane. Stations and railroad tracks are segments with ends at integer points. A station for a train with n cars is a segment of length n parallel to one of the coordinate axes. The railroad track connecting two stations is be a polyline with segments parallel to either coordinate axes, or to bisectors of angles formed by coordinate axes (i.e. run at 45° to coordinate axes). The ends of the polyline must coincide with ends of the stations to be connected (any end of each station can be connected to any end of the other one). The track is allowed to have intersections with itself or stations (Peter can always build bridges or tunnels).

In order for the train to be able to run along the track, the internal angles of the polyline must be 135° — sharp corners of 90° or 45° are not allowed. Similarly, there must be no such sharp corners between the station and the track. The train must never perform two turnings at a time, neither it should exit station and turn at the same time, or exit station and enter station at the same time, so the length of each track segment must be at least n.

Two pictures below show examples of two stations connected by a track. The variant on the right is better.





Peter has positioned the stations and now he wants to construct the shortest possible track connecting them. Help him to do it.

Input

The first line of the input file contains n ($1 \le n \le 10$). The second line contains four integer numbers: x_1, y_1, x_2 and y_2 — coordinates of the ends of the first station. The coordinates are integer, the segment $(x_1, y_1) - (x_2, y_2)$ is parallel to one of the coordinate axes and has length of exactly n. The third line contains the description of the second station in the same format. All coordinates are from 0 to 50. Stations have no common points.

Output

The first line of the output file must contain l — the number of segments of the track. The following $l+1$ lines must contain two integer numbers each — the coordinates of ends and the turning points of the track.

Examples

railroad.in	railroad.out
3 0 0 3 0 0 1 3 1	7 3 0 6 3 6 7 3 10 0 10 -3 7 -3 4 0 1

Problem J. Subtrees

Input file:	`subtrees.in`
Output file:	`subtrees.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A tree is a connected undirected graph without cycles. A subtree of a tree T is a connected subgraph of T. Depending on the structure of the tree there can be different number of subtrees.

For example, a tree that is a line with n vertices has $n(n+1)/2$ subtrees, but the tree which is a star has $2^{n-1}+n-1$ subtrees.

Similarly there can be various number of ways to select two non-overlapping subtrees of the given tree, three non-overlapping subtrees of the given tree, etc.

Finally, there are $2n-1$ ways to select $n-1$ non-overlapping subtrees regardless of the tree structure, and just 1 way to select n non-overlapping subtrees (n isolated vertices).

Given a tree with n vertices, for each k from 1 to n find the number of ways to select k non-overlapping subtrees of the given tree.

Input

The first line of the input file contains n ($2 \le n \le 100$). The following $n-1$ lines describe tree edges.

Output

Output n integer numbers — for each k from 1 to n find the number of ways to select k non-overlapping subtrees of the given tree. Each number must be output modulo 10^9.

Examples

subtrees.in	subtrees.out
5 1 2 2 3 3 4 4 5	15 35 28 9 1
5 1 2 1 3 1 4 1 5	20 38 28 9 1