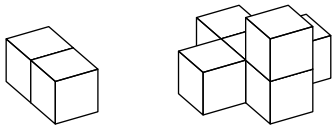
Problem A. Aztec Pyramid

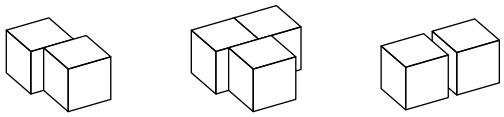
Input file:	`aztec.in`
Output file:	`aztec.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Aztec emperor Cuitláhuac is going to build a pyramid in his honor. This pyramid should be taller than previous ones.

The Aztec pyramid is build out of stone blocks. Each block is 1×1×1-hunab cube. Cuitláhuac places first block on the ground during the foundation ceremony. Each of the following blocks must share a face with at least one of the previous blocks.

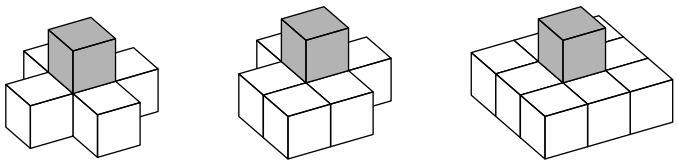


Valid block placements

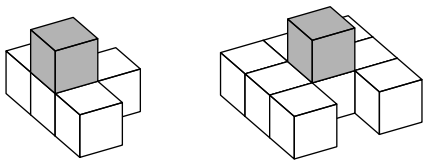


Invalid block placements

The block is stable if it stands on the ground, or it stands on another block, that has a block or the ground next to each face. To stand the test of time the pyramid must be stable i.e. each block of it must be stable.



Stable blocks



Unstable blocks

Cuitláhuac asks you to determine the height of the tallest stable pyramid that can be built out of available blocks.

Input

The only line of the input file contains a single integer number n — the number of available blocks, including the first one ($1 \le n \le 10^9$).

Output

Output the height of the tallest stable pyramid that may be built out of n blocks. The height must be output in hunabs.

Examples

aztec.in	aztec.out
6	2
5	1
20	3

Problem B. Battleship

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

You are in a computer security business. Recently one of your clients developed a new test designed to separate humans from programs. Your task is to test its efficiency.

The test is based on a variation of the Battleship game. This game is played on a ten by ten grid. Before the beginning of the test, you should place ten ships on this grid. Each ship is represented by a number of consecutive cells, arranged vertically or horizontally. You should place exactly one four-cell ship, two three-cell ships, three two-cell and four one-cell ships. No two ships should have any common or adjacent cells. They should not share a corner of a cell as well.

After the ships have been arranged, the test proceeds in a number of rounds. At each round one cell is announced to be "shot". If this cell is occupied by a ship, this ship is considered to be "hit". After each cell of a ship has been hit at least once, the ship sinks. The test ends after all the ships have sunk.

Let's call the *complexity* of an arrangement the number of round it took to finish the tests. The idea is that humans would create much more complex arrangements than programs.

You have already figured out that the cells are shot in a predefined order, each cell being shot exactly once. Now you want to write a program that would create the most complex arrangement on the basis of the order in which the cells are shot.

Input

The input file consists of ten lines, containing ten numbers each. Each number represents the round at which the corresponding cell will be shot. All numbers are distinct and belong to the range $1 \ldots 100$.

Output

Output the optimal ship arrangement for the battleship test. Empty cells should be represented by the dot ('.'), occupied cells should by represented by the number sign ('#').

Examples

battleship.in	battleship.out
1 2 3 4 5 6 7 8 9 10 36 37 38 39 40 41 42 43 44 11 35 64 65 66 67 68 69 70 45 12 34 63 84 85 86 87 88 71 46 13 33 62 83 96 97 98 89 72 47 14 32 61 82 95 100 99 90 73 48 15 31 60 81 94 93 92 91 74 49 16 30 59 80 79 78 77 76 75 50 17 29 58 57 56 55 54 53 52 51 18 28 27 26 25 24 23 22 21 20 19	...####... .... #....##... #.#.....#. ....#. ...###.... .#........ ....#. ..#.....#. .....#..#.

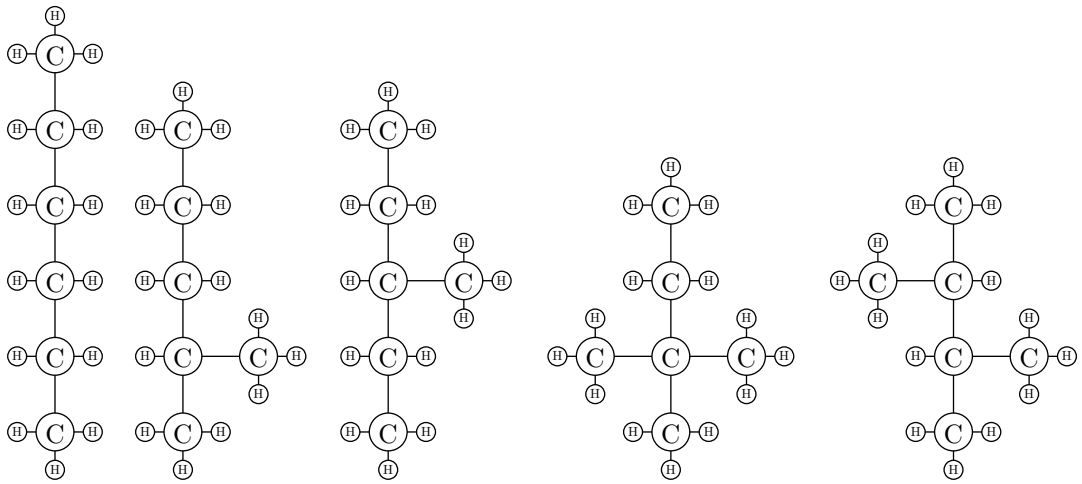
Problem C. Chemistry

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

Young Colin has recently started his organic chemistry lessons. His course has begun with the simplest organic molecules called *alkanes*.

Alkanes are molecules that consist only of hydrogen (H) and carbon (C) atoms. Each carbon atom has exactly 4 bonds (either C−H or C−C bonds), and each hydrogen atom is joined to a carbon atom (H−C bond). An alkane molecule has only single bonds and has no cycles. A molecule with n atoms of carbon always has $2n + 2$ atoms of hydrogen and is usually written as C_nH_{2n+2}.

A series of linked carbon atoms is known as the carbon skeleton. Different carbon skeletons correspond to different alkane *structural isomers*. Two isomers are considered the same if their carbon skeletons are isomorphic. Alkanes with more than three carbon atoms can be arranged in various different ways. The simplest isomer of an alkane is the one in which the n carbon atoms are arranged in a single chain with no branches. All five isomers of C_6H_{14} are shown below.



As a part of his homework Colin was asked to count the number of different alkane isomers with exactly n carbon atoms. Colin is not able to count them by himself, so he asked you for help.

Input

The only line of the input file contains a single integer number n — the number of carbon atoms in alkane molecule ($1 \le n \le 50$).

Output

The only line of output file should contain the number of different C_nH_{2n+2} isomers. It is guaranteed that the answer fits signed 64-bit integer type.

Examples

chemistry.in	chemistry.out
6	5
12	355

Problem D. Deepest Station

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

> The Saint Petersburg Metro is the underground railway system in Saint Petersburg and Leningrad Oblast, Russia. It has been open since November 15, 1955. Formerly known as the V.I. Lenin Order of Lenin Leningrad Metropoliten, the system exhibits many typical Soviet designs and features exquisite decorations and artwork making it one of the most attractive and elegant metros in the world. Due to the city's unique geology, the Saint Petersburg Metro is one of the deepest subway systems in the world and the deepest by the average depth of all the stations. The system's deepest station, Admiralteyskaya, is 105 metres below the ground. Serving two and a half million passengers daily, it is also the 12th busiest subway system in the world.
>
> From Wikipedia, the free encyclopedia

After building *Admiralteyskaya* metro station, the government of *Saint Petersburg* decided to build the really deep station which would be the deepest in the world. It will be d meters under the ground! The station will be built right under the *Smolny Sobor* and for use by officials only. The Department of Urban Development has its internal coordinate system for building a project. The origin of this system point is exactly the Smolny Sobor and applicate means depth under the ground. So, the new station will have coordinates $(0, 0, d)$.

Due to security reasons station's lobby must be located outside the *Smolny Convent*, at point (x, y). Your task is to help the government with building moving staircases. The station will use innovative staircases that should go down at angle 45°. It is possible to build one intermediate lobby somewhere underground (like at Admiralteyskaya) and two staircases. Given x, y and d, find coordinates of intermediate lobby. Note that you cannot dig below d meters under the ground, so the intermediate lobby must not be deeper than the main station.

Input

The only line of the input file contains three integer numbers: x, y and d — the coordinates of the station in the Department of Urban Development coordinate system ($-10\,000 \le x, y \le 10\,000$; $(x, y) \ne (0, 0)$; $106 \le d \le 10\,000$).

Output

Output three numbers with precision at least 10 digits after decimal point: coordinates of the intermediate lobby in the Department of Urban Development coordinate system.

If it is impossible to build staircases, output "`Impossible`". If no intermediate lobby is required, output "`Single staircase`".

Examples

deepest.in	deepest.out
0 100 300	0.0 200.0 100.0
300 400 500	Single staircase
400 400 500	Impossible

Problem E. Electricity

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

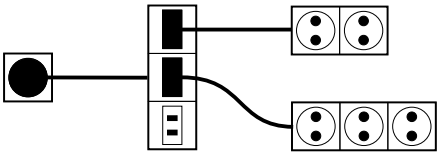
You are managing a power network for a programming competition and you have to connect a lot of computers to the power supply. Unfortunately, there are two standards for electrical plug and socket: A and B. These standards are incompatible, so the plug of standard A can only be plugged in the socket of standard A and the plug of standard B can only be plugged in the socket of standard B.

In the main competition hall, there is exactly one main socket of type A. Every computer that will be used in this programming competition has one plug of type A. Thereby only one computer can be connected directly to the main socket. But you have a number of power strips of two types.

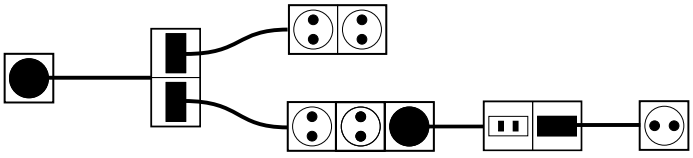
- Power strips of the first type have one plug of type A and several sockets of type B.
- Power strips of the second type have one plug of type B and several sockets of type A.

All the power strips are very powerful and can withstand any load. So you can create a power network by connecting one power strip of the first type to the main socket, then some power strips of the second type to this power strip of the first type, etc. At the end you will get several available sockets of type A for computers.

Your task is to find the maximum number of computers that can be connected to the power network, using available power strips.



Possible solution for the first example.



Possible solution for the second example.

Input

The first line of input file contains two integer numbers n and m — the number of power strips of the first and the second type ($0 \le n, m \le 100\,000$).

The second line contains n integer numbers a_i — the number of sockets on the power strips of the first type ($1 \le a_i \le 1000$).

The second line contains m integer numbers b_i — the number of sockets on the power strips of the second type ($1 \le b_i \le 1000$).

Output

Output the maximum number of computers that can be connected to the power network.

Examples

electricity.in	electricity.out
3 2 3 2 1 2 3	5
2 3 2 2 2 3 1	5

Problem F. Final Standings

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

Formula TheByte is the most famous race competition in *ByteLand*. The competition is over and each of n drivers has a number of points. A driver that has more points is placed higher.

Final standings are not disclosed yet, but we know that the total number of points earned by all the drivers is p and there are only d different numbers of points among top k drivers.

The *ByteLand Times* asks you to guess final standings based on the given information.

Input

The only line of the input file contains four integer numbers: n, p, k and d — the number of drivers, the number of points, the number of top drivers, and the number of different numbers of points among top k drivers ($1 \le k \le n \le 1000$; $0 \le p \le 1\,000\,000$; $1 \le d \le k$).

Output

Output the possible standings that match given n, p, k and d.

If it is possible to create the correct final standings, you should output final standings. The i-th line of the final standings must contain a number of points earned by i-th driver. Drivers should be ordered by number of points in the descending order.

If there is no possible final standings satisfying the given data, output the single line "`Wrong information`".

Examples

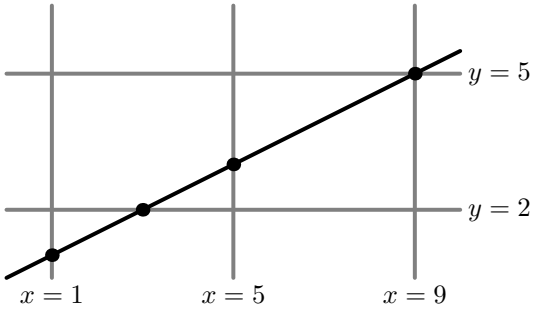
final.in	final.out
3 4 2 2	2 1 1
3 5 2 2	3 2 0
2 5 2 1	Wrong information

Problem G. Grid

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

Gerald has spent several hours drawing a rectangular grid on a sheet of paper. At the beginning, he drew some vertical lines with equal distance (d_x) between them. Than he drew some horizontal lines, also with equal distance (d_y) between them. Both d_x and d_y are greater than zero.

While Gerald was relaxing with a cup of tea, his brother Mike came and scratched a straight line on the sheet. Gerald felt outraged with Mike's behavior and ordered him to remove everything odd from the paper.



Mike didn't take his brother's words seriously and removed almost all the information with an eraser. However, he did not notice the points of intersection of his line with the grid. All those points were bold enough to be readable even after erasing.

Help Gerald to find the parameters of the original grid.

Input

The first line of input contains single integer n — the number of points of intersection ($3 \le n \le 100\,000$).

Each of the following n lines contains a pair of integer number x_i, y_i — the coordinates of the intersection point. Coordinates do not exceed 10^9 by the absolute value.

All the intersection points are distinct. There are no common points of the grid and the Mike's line except for the specified ones.

Output

Output six integer numbers x_1, x_2, d_x, y_1, y_2 and d_y. First three numbers describe the set of vertical lines: the minimum x-coordinate of the vertical line, the maximum x-coordinate of the vertical line, and the distance between adjacent vertical lines ($-10^9 \le x_1 \le x_2 \le 10^9$; $0 < d_x \le 2 \cdot 10^9$). Following three numbers should describe the set of horizontal lines in the same way with x replaced by y ($-10^9 \le y_1 \le y_2 \le 10^9$; $0 < d_y \le 2 \cdot 10^9$).

It is guaranteed that at least one solution exists.

Examples

grid.in	grid.out
4 1 1 5 3 3 2 9 5	1 9 4 2 5 3

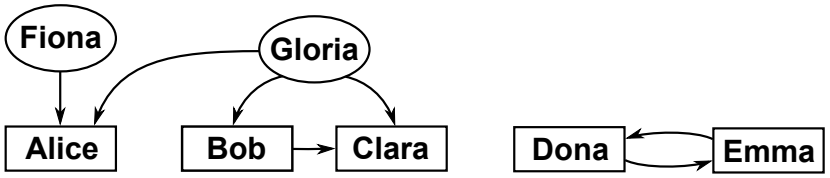
Problem H. Hospital

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

You are writing a personnel management tool for a large hospital. One of the problems is managing the vacation plans of employees.

There are two types of nurses working in the hospital: *staff nurses* continuously take care of the patients and *special nurses* fill special positions like "surgical nurse" or "nurse supervisor". When a staff nurse goes on vacation, nothing critical happens, because other nurses will do her work. But when special nurse goes on vacation, she must be substituted. For every special nurse there is a list of nurses, who can substitute her. If the substituting nurse is a special nurse too, then some third nurse must substitute her as well, and so on. Sometimes it is impossible for special nurse to go on vacation, because there is no sequence of substitutions, after that all special nurse positions are filled.

Consider the following example. There are seven nurses in a hospital. Alice, Bob, Clara, Dona and Emma are special nurses, Fiona and Gloria are staff nurses. They can substitute each other according to the following scheme (arrow from A to B means A can substitute B):



In this example, Dona and Emma cannot go on vacation, because if one of them does, another cannot fill both positions. Alice, Bob and Clara can go on vacation, but Bob and Clara cannot do it at the same time, because Gloria cannot substitute them both simultaneously.

Your task is to find all nurses who cannot go on vacation, and all pairs of nurses, such that both can go on vacation, but not in the same time.

Input

The first line of the input file contains two integer numbers n and k — the total number of nurses and number of special nurses. Nurses $1 \dots k$ are special nurses, and nurses $k+1 \dots n$ are staff nurses ($1 \le k < n \le 1000$). Next k lines contain the lists of possible substitutions. The first number in $(i+1)$-th line is d_i — the number of nurses, who can substitute i-th nurse. Following d_i numbers in the same line are the indices of substituting nurses. The total length of all lists does not exceed 10 000.

Output

First output the number of nurses who cannot go on vacation. Then output their indices.

After that, output the number of pairs of nurses, such that both can go on vacation, but not at the same time. Finally, if the number of such pairs is not greater then 10 000, output all these pairs (pairs (A, B) and (B, A) are considered equal).

Examples

hospital.in	hospital.out
7 5 2 6 7 1 7 2 2 7 1 5 1 4	2 4 5 3 2 3 2 7 3 7

Problem I. Intelligent Design

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

You probably know about artificial neural networks and their learning algorithms. Sometimes people even believe that neural networks can magically solve any problem "by themselves". However, in this problem we will treat neural networks as a simple computation model. You are asked to write a program, intelligently designing a neural network computing a given logical formula.

The logical formula may contain input variables from A to L, constants 0 and 1, combined with logical operators: equivalence ($\Leftrightarrow$), implication ($\rightarrow$), disjunction ($\vee$), conjunction ($\wedge$), and negation ($\neg$) according to the following grammar:

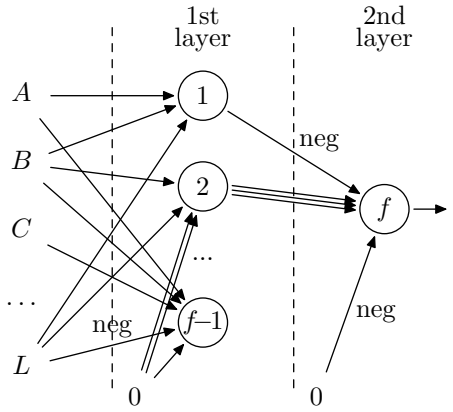
$$\begin{aligned}
\langle\text{expression}\rangle &::= \{\langle\text{implication}\rangle \Leftrightarrow\}^* \langle\text{implication}\rangle \\
\langle\text{implication}\rangle &::= \langle\text{disjunction}\rangle \mid \langle\text{disjunction}\rangle \rightarrow \langle\text{implication}\rangle \\
\langle\text{disjunction}\rangle &::= \langle\text{conjunction}\rangle \mid \langle\text{disjunction}\rangle \vee \langle\text{conjunction}\rangle \\
\langle\text{conjunction}\rangle &::= \langle\text{term}\rangle \mid \langle\text{conjunction}\rangle \wedge \langle\text{term}\rangle \\
\langle\text{term}\rangle &::= A \ldots L \mid 0 \mid 1 \mid \neg \langle\text{term}\rangle \mid (\langle\text{expression}\rangle)
\end{aligned}$$

Here $\{X\}^*$ means repetition of X zero or more times.

The semantics of operations is traditional, except for equivalence. Equivalence of several arguments written in a row inside the same expression is 1 when and only when all arguments have equal values.

For this problem we will use the following neural network model. The signals are binary (either 0 or 1). Each network node is a *majority* element — its *axon* (output) is 1 then and only then, when more than a half of its *dendrites* (inputs) are 1. For example, if some node has four dendrites, then its axon switches to 1 when at least three dendrites are set to 1. Each node may be programmed to negate some of its dendrites before computing majority function.

The network must contain two layers. If f is the number of its nodes, then $f-1$ nodes (indexed from 1 to $f-1$) must belong to the first layer and the last node (number f) — to the second layer. The function inputs may be connected to the first layer dendrites only, the first layer axons may be connected to the second layer dendrites only, the second layer axon is the network output. Any dendrite may also be connected to constant 0.



This fragment of some neural network demonstrates various connection types. Here input A is connected to nodes 1 and $f-1$, inputs B and L are connected to all first layer nodes, while input C connected only to node $f-1$. Constant zero is connected to node 2 twice, so this node has four dendrites; similarly, node 2 is connected to node f three times. Axon of node $f-1$ is not connected. Connections of input L to node $f-1$, node 1 to node f, and constant zero to node f are negated.

Input

The only line of the input file contains a logical formula (not longer than 300 000 symbols). Logical operations are represented by '`<=>`' (equivalence), '`->`' (implication), '`|`' (disjunction), '`&`' (conjunction), and '`~`' (negation). The variables '`A`' ... '`L`' and constants '`0`' and '`1`' stand for themselves. The formula contains no spaces or other symbols not mentioned in the grammar.

Output

The first line of the output file must contain f — the number of nodes ($2 \le f \le 5000$). The following f lines describe connections of the network, a line per node, in the order of the node's indexes.

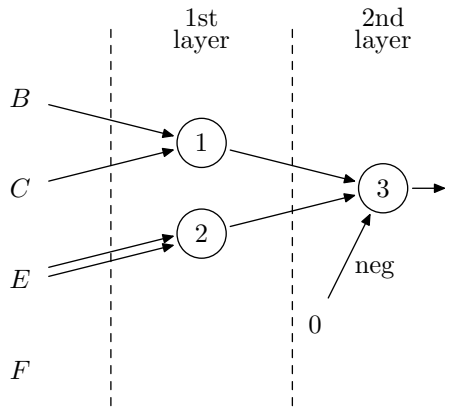
Each node description line contains the number of the node dendrites D_k ($1 \le D_k \le 5000$), followed by D_k values, explaining connection of each dendrite. Each value is either 0 (for zero constant), or input variable name, or integer from 1 to $f - 1$ (standing for axon of the corresponding 1st layer node). The value may be preceded with tilde ('`~`'), standing for negation of the corresponding dendrite signal.

Examples

intelligent.in	intelligent.out

`(B&C|E)&(F<=>(F)<=>F)`	`3`
	`2 B C`
	`2 E E`
	`3 1 2 ~0`

The solution is visualized in the picture below.



Here two dendrites of node 2 are connected to the same input E. Node 3 is connected to two first layer nodes 1 and 2 and to constant zero. The zero connection is negated. Input F is not connected to any node.

Problem J. Juggle with Criteria

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

A *permutation* p of length n is an array $\{p_1, p_2, \ldots, p_n\}$ that contains each integer from 1 to n once.

Jack devised five criteria that are supposed to show how close a permutation p is to the *identity permutation* $\{1, 2, \ldots, n\}$:

$a(p)$ = the number of inversions in p (pairs of indices (i, j) such that $i < j$ and $p_i > p_j$);
$b(p)$ = the number of local inversions in p (indices i such that $p_i > p_{i+1}$);
$c(p)$ = the length of longest increasing subsequence in p
($p_{i_1} < p_{i_2} < \ldots < p_{i_k}$ with increasing indices $i_1 < i_2 < \ldots < i_k$);
$d(p)$ = the length of longest increasing substring in p
(increasing consecutive subsequence $p_i < p_{i+1} < \ldots < p_{j-1} < p_j$);
$e(p)$ = the number of fixed points (indices i such that $p_i = i$).

Jill wants to prove him that these criteria can vary independently (in some sense). She need to show that two permutations p and q of equal length can give any combination of:

whether $a(p)$ is less, equal, or greater than $a(q)$;
whether $b(p)$ is less, equal, or greater than $b(q)$;
whether $c(p)$ is less, equal, or greater than $c(q)$;
whether $d(p)$ is less, equal, or greater than $d(q)$; and
whether $e(p)$ is less, equal, or greater than $e(q)$.

Help Jill prove her thesis in a constructive way. For a given set of relations between these criteria, find two permutations p and q of the same length that satisfy these relations.

Input

The first line of the input file contains two integer numbers n and l — the number of relations and permutations length ($1 \le n \le 243$; $1 \le l \le 1000$).

Each of the following n lines contains one set criteria that is defined by the five characters. Each character is either '`<`', '`=`', or '`>`'. These characters denote desired relations between $a(p)$ and $a(q)$, ..., $e(p)$ and $e(q)$ (in this order).

Output

For each set of criteria given in the input file output either "`Exists`" if there exists a pair of permutations p and q of length l satisfying this set of criteria, and "`Not exists`" otherwise.

In the former case following two lines must contain permutation p and q (in this order).

Examples

juggle.in	juggle.out
3 4 <==> <<<< =====	Exists 1 4 2 3 2 3 4 1 Not exists Exists 1 2 3 4 1 2 3 4

In the first pair of permutations in sample output:
$a(p)=2 < 3=a(q)$; $b(p)=1 = 1=b(q)$; $c(p)=3 = 3=c(q)$; $d(p)=2 < 3=d(q)$; $e(p)=1 > 0=e(q)$.

Problem K. Kingdom Subdivision

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

Once upon a time there was a kingdom ruled by a wise king. After forty years of his reign, by means of successful military actions and skillful diplomacy, the kingdom became a polygon with n vertices without self-intersections, self-touches and holes. This form of the kingdom greatly simplified tax collection and land division between the inhabitants, so everybody was happy.

But nothing lasts forever, and one autumn evening the king suddenly died. Due to the laws of the kingdom, the oldest son of the king should have taken the kingdom, but the problem was that he had two twin sons. It was widely known that the king loved his sons equally, so it was not even pronounced to choose only one son to rule.

After ten days of serious thoughts it was finally decided to split the kingdom in two parts — one per each son. These parts should, of course, have equal area. But as well, as everybody got used to simplicity of land division and tax accounting, every part should be a polygon without self-intersections, self-touches and holes. A great problem for great thinkers! For three days and three nights magicians, astrologers and alchemists of the kingdom tried to divide the kingdom, but everything they invented reduced to at least #P computation problems. So they decided to look into the future and called you to solve this problem.

Input

The first line of the input file contains single integer number n — the number of vertices in the kingdom ($3 \le n \le 5000$). Each of the following n lines contains two integers x_i, y_i — the coordinates of the i-th vertex ($0 \le |x_i|, |y_i| \le 10^6$). The vertices are given in the counterclockwise order. No three consecutive vertices lie on the same line. The kingdom does not have self-intersections, self-touches and holes.

Output

Output the descriptions of the two kingdom parts. Each description should follow the format given above, except that the coordinates can be real-valued. The number of vertices in each part must not exceed 20 000. The interiors of the parts must not have common points, and their union must be equal to the original kingdom. Each part should not have self-intersections, self-touches or holes. Equal consecutive vertices and three or more collinear vertices in a row are allowed.

Let the areas of the parts be A and B. The answer is considered to be correct if $\frac{|A-B|}{\max(A,B)} \le 10^{-3}$.

Note that the checker considers points P and Q equal if $|P.x - Q.x| \le 10^{-9}$ and $|P.y - Q.y| \le 10^{-9}$. Also, it considers that the point P lies on a segment S if the distance between P and Q does not exceed 10^{-9}.

Examples

kingdom.in	kingdom.out
4 0 0 -1 -1 1 0 -1 1	3 0 0 -1 -1 1 0 3 0 0 1 0 -1 1

Problem L. Log Analysis

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

Lisa writes a log analysis application for a distributed computer system. Unlike single-node logs, that are append-only the distributed log is highly volatile. When a node becomes online, it may push a batch of events in the past of the log. Conversely, when it goes offline some log entries may disappear.

To ensure the stability and availability of the application Lisa need to monitor the number of distinct events in the log segments. She is going to handle distributed part, while you have to implement a local one.

Your program is started from the empty log and must support the following operations:

- **insert** $\langle index \rangle$ $\langle number \rangle$ $\langle type \rangle$ — inserts $\langle number \rangle$ of events of type $\langle type \rangle$ before event with index $\langle index \rangle$. Events with indices larger or equal to $\langle index \rangle$ are renumbered.
- **remove** $\langle index \rangle$ $\langle number \rangle$ — removes $\langle number \rangle$ of events starting from event with index $\langle index \rangle$.
- **query** $\langle index_1 \rangle$ $\langle index_2 \rangle$ — counts the number of distinct event types for events with indices from $\langle index_1 \rangle$ to $\langle index_2 \rangle$ inclusive.

The events are indexed starting from 1. The event types are represented by single-letter codes.

Input

The first line of the input file contains single integer number n — the number of operations ($1 \le n \le 30\,000$). The following n lines contain one operation description each.

Operation description starts with operation type: '`+`' for insert, '`-`' for remove and '`?`' for query. Operation type is followed by operation arguments.

All indices are valid, i. e. events with specified indices exist, and you never have to remove events past the end of the log.

The $\langle number \rangle$ for the insert and remove operations does not exceed 10 000.

Event types are represented by lowercase Latin letter.

Output

For each query operation output a single number — the number of distinct event types between $\langle index_1 \rangle$ and $\langle index_2 \rangle$ inclusive.

Examples

log.in	log.out
8 + 1 4 w + 3 3 o ? 2 3 - 2 2 ? 2 3 + 2 2 t ? 1 6 - 1 6	2 1 3