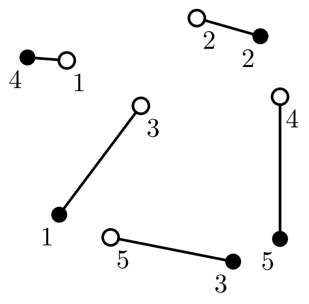
Problem A. Ants

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

Young naturalist Bill studies ants in school. His ants feed on plant-louses that live on apple trees. Each ant colony needs its own apple tree to feed itself.

Bill has a map with coordinates of n ant colonies and n apple trees. He knows that ants travel from their colony to their feeding places and back using chemically tagged routes. The routes cannot intersect each other or ants will get confused and get to the wrong colony or tree, thus spurring a war between colonies.

Bill would like to connect each ant colony to a single apple tree so that all n routes are non-intersecting straight lines. In this problem such connection is always possible. Your task is to write a program that finds such connection.



On this picture ant colonies are denoted by empty circles and apple trees are denoted by filled circles. One possible connection is denoted by lines.

Input

The first line of the input file contains a single integer number n ($1 \leq n \leq 100$) — the number of ant colonies and apple trees. It is followed by n lines describing n ant colonies, followed by n lines describing n apple trees. Each ant colony and apple tree is described by a pair of integer coordinates x and y ($-10\,000 \leq x, y \leq 10\,000$) on a Cartesian plane. All ant colonies and apple trees occupy distinct points on a plane. No three points are on the same line.

Output

Write to the output file n lines with one integer number on each line. The number written on i-th line denotes the number (from 1 to n) of the apple tree that is connected to the i-th ant colony.

Sample input and output

ants.in	ants.out

```
5
-42 58
44 86
7 28
99 34
-13 -59
-47 -44
86 74
68 -75
-68 60
99 -60
```
	```
	4
	2
	1
	5
	3
	```

Problem B. Building for UN

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

The United Nations has decided to build a new headquarters in Saint Petersburg, Russia. It will have a form of a rectangular parallelepiped and will consist of several rectangular floors, one on top of another. Each floor is a rectangular grid of the same dimensions, each cell of this grid is an office.

Two offices are considered adjacent if they are located on the same floor and share a common wall, or if one's floor is the other's ceiling.

The St. Petersburg building will host n national missions. Each country gets several offices that form a connected set.

Moreover, modern political situation shows that countries might want to form secret coalitions. For that to be possible, each pair of countries must have at least one pair of adjacent offices, so that they can raise the wall or the ceiling they share to perform secret pair-wise negotiations just in case they need to.

You are hired to design an appropriate building for the UN.

Input

The input file consists of a single integer number n ($1 \leq n \leq 50$) — the number of countries that are hosted in the building.

Output

On the first line of the output file write three integer numbers h, w, and l — height, width and length of the building respectively.

h descriptions of floors should follow. Each floor description consists of l lines with w characters on each line. Separate descriptions of adjacent floors with an empty line.

Use capital and small Latin letters to denote offices of different countries. There should be at most 1 000 000 offices in the building. Each office should be occupied by a country. There should be exactly n different countries in the building. In this problem the required building design always exists.

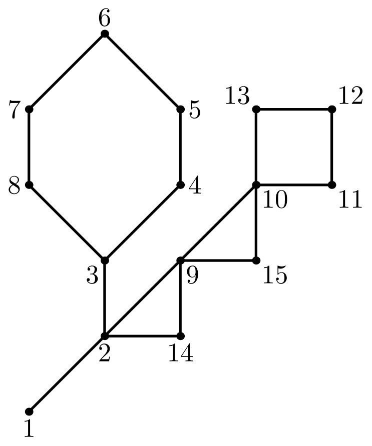
Sample input and output

`building.in`	`building.out`
`4`	`2 2 2`
	`AB`
	`CC`
	`zz`
	`zz`

Problem C. Cactus Reloaded

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

Cactus is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively cactus is a generalization of a tree where some cycles are allowed. Your task is to find a *diameter* of the given cactus. Diameter is the maximal length of the shortest path between pairs of vertices.



For example, on the picture above the shortest path between vertices 6 and 12 goes through 8 edges and it is the maximal shortest path in this graph, thus its diameter is 8.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n \le 50\,000$, $0 \le m \le 10\,000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer number k_i ($2 \le k_i \le 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. Adjacent vertices in a path are distinct. Path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

The graph in the input file is a cactus.

Output

Write to the output file a single integer number — the diameter of the given cactus.

Sample input and output

`cactus.in`	`cactus.out`

`15 3`
`9 1 2 3 4 5 6 7 8 3`
`7 2 9 10 11 12 13 10`
`5 2 14 9 15 10`
	`8`

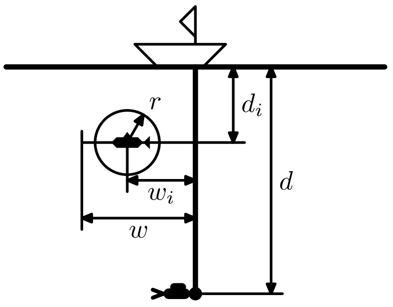
Problem D. Diver

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

Diver had just completed her mission in the depths of the ocean and needs to resurface. To get to the surface she must use the rope that goes straight down from her boat on the surface to her location d feet under the water. However, while she was working, several sharks gathered near the rope. They do not consider her a danger or a prey yet, but if she gets closer than r feet to a shark, then it immediately attacks her.

To avoid decompression sickness diver can descend (go down) or ascend (go up) at most v_d feet per second. She also cannot go deeper than d feet under the water.

Each shark swims at its own constant depth of d_i feet near the rope. Speed and the pattern of movement for all sharks is the same. They cannot just stay in the water near the rope. They have to swim to avoid sinking, so they swim in a back-and-forth motion with a constant speed of v_s — swimming away from the rope on distance of w feet and swimming back to the rope again. Sharks change the direction of their movement so fast, that we consider it being instantaneous. When a shark attacks the diver it also moves so fast, that we consider it to happen instantaneously as soon as the diver is inside a circle of r feet in radius around a shark.



Your task is to figure out if the diver can get to the surface without being attacked by a shark, and if yes, then how fast she can do it.

Input

The first line of the input file contains 6 integer numbers:

- d ($10 \le d \le 100$) — initial depth of the diver.
- v_d ($1 \le v_d \le 10$) — maximal speed of the diver.
- n ($1 \le n \le 20$) — number of sharks.
- r ($1 \le r \le 10$) — minimal safe distance between a shark and the diver.
- w ($10 \le w \le 100$) — maximal distance that a shark swims away from the rope.
- v_s ($1 \le v_s \le 50$) — speed of a shark.

Then follow n lines describing sharks with 3 integer numbers per line for each shark:

- d_i ($1 \le d_i < d$) — depth of i-th shark.
- w_i ($0 \le w_i \le w$) — initial distance from i-th shark to the rope.
- f_i (f_i is 1 or -1) — initial direction of i-th shark's movement in relation to the rope (1 if it swims away from the rope, or -1 if it swims to the rope).

Initially the diver is more than r feet from any shark.

Output

Write to the output file `IMPOSSIBLE` if the diver cannot get to the surface or write the minimal time that it will take the diver to resurface with precision of at least 10^{-5}.

Sample input and output

diver.in	diver.out

`10 1 2 1 10 1`
`6 4 -1`
`1 1 1`
	`11.414213562373096`

Problem E. Equation

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

Your task is to solve an equation of the form $f(x) = 0$ where $f(x)$ is written in postfix notation with numbers, operations `+`, `-`, `*`, `/`, and at most one occurrence of a variable x.

For example, $f(x)$ for an equation $(4x + 2)/2 = 0$ is written as:

```
4 X * 2 + 2 /
```

The solution for $f(x) = 0$ is $x = -1/2$.

Input

The input file consists of a single line with at most 30 tokens separated by spaces. Each token is either:

- a digit from `0` to `9`;
- an operation `+`, `-`, `*`, or `/`;
- an uppercase letter `X` that denotes variable x.

The input file contains a correct representation of $f(x)$ in postfix notation where token `X` occurs at most once. There is no division by a constant zero in this equation, that is, there always exists a value of x, such that $f(x)$ can be evaluated without division by zero.

Output

Write to the output file:

- `X =` p/q if equation $f(x) = 0$ has a single solution that can be represented with a simple fraction p/q, where p and q are coprime integer numbers and q is positive.
- `NONE` if equation $f(x) = 0$ has no solution;
- `MULTIPLE` if equation $f(x) = 0$ has multiple solutions.

Sample input and output

equation.in	equation.out
`4 X * 2 + 2 /`	`X = -1/2`
`2 2 *`	`NONE`
`0 2 X / *`	`MULTIPLE`

Problem F. Fund Management

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

Frank is a portfolio manager of a closed-end fund for *Advanced Commercial Markets* (*ACM*). Fund collects money (cash) from individual investors for a certain period of time and invests cash into various securities in accordance with fund's investment strategy. At the end of the period all assets are sold out and cash is distributed among individual investors of the fund proportionally to their share of original investment.

Frank manages equity fund that invests money into stock market. His strategy is explained below.

Frank's fund has collected c US Dollars (USD) from individual investors to manage them for m days. Management is performed on a day by day basis. Frank has selected n stocks to invest into. Depending on the overall price range and availability of each stock, a *lot size* was chosen for each stock — the number of shares of the stock Frank can buy or sell per day without affecting the market too much by his trades. So, if the price of the stock is p_i USD per share and the lot size of the corresponding stock is s_i, then Frank can spend $p_i s_i$ USD to buy one lot of the corresponding stock for his fund if the fund has enough cash left, thus decreasing available cash in the fund. This trade is completely performed in one day.

When price of the stock changes to p'_i later, then Frank can sell this lot for $p'_i s_i$ USD, thus increasing available cash for further trading. This trade is also completely performed in one day. All lots of stocks that are held by the fund must be sold by the end of the fund's period, so that at the end (like at the beginning) the fund is holding only cash.

Each stock has its own volatility and risks, so to minimize the overall risk of the fund, for each stock there is the maximum number of lots k_i that can be held by the fund at any given day. There is also the overall limit k on the number of lots of all stocks that the fund can hold at any given day.

Any trade to buy or sell one lot of stock completely occupies Frank's day, and thus he can perform at most one such trade per day. Frank is not allowed to buy partial lots if there is not enough cash in the fund for a whole lot at the time of purchase.

Now, when fund's period has ended, Frank wants to know what is the maximum profit he could have made with this strategy having known the prices of each stock in advance. Your task is to write a program to find it out.

It is assumed that there is a single price for each stock for each day that Frank could have bought or sold shares of the stock at. Any overheads such as fees and commissions are ignored, and thus cash spent to buy or gained on a sell of one lot of stock is exactly equal to its price on this day multiplied by the number of shares in a lot.

Input

The first line of the input file contains four numbers — c, m, n, and k. Here c ($0.01 \le c \le 100\,000\,000.00$) is the amount of cash collected from individual investors up to a cent (up to two digits after decimal point); m ($1 \le m \le 100$) is the number of days in the fund's lifetime; n ($1 \le n \le 8$) is the number of stocks selected by Frank for trading; k ($1 \le k \le 8$) is the overall limit on the number of lots the fund can hold at any time.

The following $2n$ lines describe stocks and their prices with two lines per stock.

The first line for each stock contains the stock name followed by two integer numbers s_i and k_i. Here s_i ($1 \le s_i \le 1\,000\,000$) is the lot size of the given stock, and k_i ($1 \le k_i \le k$) is the number of lots of this stock the fund can hold at any time. Stock name consists of 1 to 5 capital Latin letters from "A" to "Z". All stock names in the input file are distinct.

The second line for each stock contains m decimal numbers separated by spaces that denote prices of the corresponding stock for each day in the fund's lifetime. Stock prices are in range from 0.01 to 999.99

(inclusive) given up to a cent (up to two digits after decimal point).

Cash and prices in the input file are formatted as a string of decimal digits, optionally followed by a dot with one or two digits after a dot.

Output

Write to the output file $m + 1$ lines.

On the first line write a single decimal number — the precise value for the maximal amount of cash that can be collected in the fund by the end of its period. The answer will not exceed 1 000 000 000.00. Cash must be formatted as a string of decimal digits, optionally followed by a dot with one or two digits after a dot.

On the following m lines write the description of Frank's actions for each day that he should have made in order to realize this profit. Write `BUY` followed by a space and a stock name for buying a stock. Write `SELL` followed by a space and a stock name for selling a stock. Write `HOLD` if nothing should have been done on that day.

Sample input and output

fund.in

```
144624.00 9 5 3
IBM 500 3
97.27 98.31 97.42 98.9 100.07 98.89 98.65 99.34 100.82
GOOG 100 1
467.59 483.26 487.19 483.58 485.5 489.46 499.72 505 504.28
JAVA 1000 2
5.54 5.69 5.6 5.65 5.73 6 6.14 6.06 6.06
MSFT 250 1
29.86 29.81 29.64 29.93 29.96 29.66 30.7 31.21 31.16
ORCL 300 3
17.51 17.68 17.64 17.86 17.82 17.77 17.39 17.5 17.3
```

fund.out

```
151205.00
BUY GOOG
BUY IBM
BUY IBM
HOLD
SELL IBM
BUY MSFT
SELL MSFT
SELL GOOG
SELL IBM
```

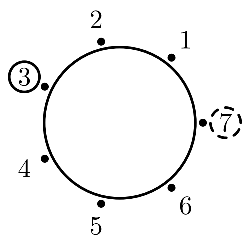
Problem G. Game

Input file: game.in
Output file: game.out

A group of contestants sits at the round table and plays the following game to relieve anxiety before the start of NEERC 2007. The game is played with a single token that is given to one person at the beginning of the game. This person passes the token to the adjacent person on the left-hand side or to the adjacent person on the right-hand side with a certain probability. A person who receives the token does the same with his own probability and so on. The game ends when each person has received the token at least once. The last person who has received the token wins.

The problem is to find the probability of winning for the given person. The probability of passing the token to the left or to the right is individual for each person and is known in advance before the beginning of the game.

Contestants are numbered from 1 to n so that the person number 2 sits to the right of 1, the person number 3 sits to the right of 2, and so on. The person number 1 sits to the right of n. The game starts with the person whose number is specified in the input file and your task is to find the probability of winning for the person number n.



Picture shows 7 contestants at the table with the token given to the person number 3.

Input

The first line of the input file contains two integer numbers n and k ($2 \le n \le 50$, $1 \le k < n$). n denotes the total number of contestants, k denotes the number of the person who has the token at the beginning of the game.

The second line of the input file contains $n-1$ numbers that denote the probabilities p_i ($0.01 \le p_i \le 0.99$) of passing the token to the right for the persons numbered from 1 to $n-1$. The probability of passing the token to the left for the person number i is $1 - p_i$. The probabilities are given with at most 2 digits after decimal point.

Output

Write to the output file a single number that denotes the probability of winning for the person number n with a precision of at least 6 digits after decimal point.

Sample input and output

game.in	game.out
7 3 0.5 0.5 0.5 0.5 0.5 0.5	0.1666666667
3 1 0.3 0.6	0.3000000000

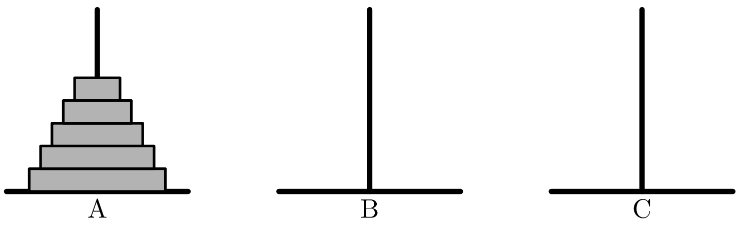
24 12
0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.5
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
	0.9800000000

Note: all probabilities in the third example are on the same line in the actual input file.

Problem H. Hanoi Towers

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

The “Hanoi Towers” puzzle consists of three pegs (that we will name A, B, and C) with n disks of different diameters stacked onto the pegs. Initially all disks are stacked onto peg A with the smallest disk at the top and the largest one at the bottom, so that they form a conical shape on peg A.



A valid move in the puzzle is moving one disk from the top of one (source) peg to the top of the other (destination) peg, with a constraint that a disk can be placed only onto an empty destination peg or onto a disk of a larger diameter. We denote a move with two capital letters — the first letter denotes the source disk, and the second letter denotes the destination disk. For example, `AB` is a move from disk A to disk B.

The puzzle is considered solved when all the disks are stacked onto either peg B (with pegs A and C empty) or onto peg C (with pegs A and B empty). We will solve this puzzle with the following algorithm.

All six potential moves in the game (`AB`, `AC`, `BA`, `BC`, `CA`, and `CB`) are arranged into a list. The order of moves in this list defines our strategy. We always make the first valid move from this list with an additional constraint that we never move the same disk twice in a row.

It can be proven that this algorithm always solves the puzzle. Your problem is to find the number of moves it takes for this algorithm to solve the puzzle using a given strategy.

Input

The input file contains two lines. The first line consists of a single integer number n $(1 \leq n \leq 30)$ – the number of disks in the puzzle. The second line contains descriptions of six moves separated by spaces — the strategy that is used to solve the puzzle.

Output

Write to the output file the number of moves it takes to solve the puzzle. This number will not exceed 10^{18}.

Sample input and output

hanoi.in	hanoi.out
3 AB BC CA BA CB AC	7
2 AB BA CA BC CB AC	5

Problem I. I18n

Input file:	`i18n.in`
Output file:	`i18n.out`

Internationalization and localization are long words that are usually abbreviated as i18n and l10n. The numbers in between i-n and l-n refer to the number of letters that were omitted. It is a very powerful abbreviation method that can be naturally used for any words.

A word may be abbreviated only when it has previously occurred in a given text. A word is abbreviated by omitting all the letters in the word except for the first and last letter and replacing omitted letters with a number of omitted letters.

Your task is to write a program that expands such abbreviations in a given text whenever possible. Expansion is possible if it is *valid* and *unambiguous*.

Expansion is valid if expanded word has previously occurred in the text and its abbreviated form corresponds to the abbreviation that is being expanded. Case is ignored for the purposes of validness. For example, expansion from i18n to internationalization is valid in this problem statement (even as internationalization was previously mentioned only with capital letter I). But expansion of p14n to parameterization is not valid since the word parameterization has never occurred before its abbreviation, and expansion of a11n to abbreviation is not valid, since it is not a correct abbreviation for the word abbreviation (correct one is a10n).

Expansion is unambiguous if there is exactly one valid expansion for it. For example, expansion from l10n to localization is unambiguous in this problem statement, but expansion from p5m cannot be made unambiguously, since both problem and program are abbreviated to p5m.

Input

The input file contains at most 1000 lines with at most 80 characters in each line. Each line contains one or more words separated by spaces and special symbols: '`-`', '`,`', '`.`', '`"`', '`(`', '`)`', '`:`', '`;`', '`!`', '`?`'. There are no trailing spaces, but other separators are allowed at the end of line.

Words may be either *full* or *abbreviated*. Full word consists of one to 32 Latin letters and may be written in one of three *capitalization* forms: all lowercase, First Capital Letter, or ALL CAPITAL LETTERS. Abbreviated word consists of a Latin letter, followed by a number from 2 to 30 (no leading zero), followed by a Latin letter. Abbreviated words also have three corresponding capitalization forms: all l7e, F3t C5l L4r, or ALL C5L L5S.

Output

Write to the output file original text with original separators while expanding abbreviated words into full words whenever possible (see problem statement). Capitalization of the expanded full word shall correspond to the capitalization of the abbreviation that is being expanded.

Invalid or ambiguous abbreviations shall be left in the text as is (abbreviated). Note, that lines in the output file may be longer than 80 characters.

Sample input and output

i18n.in	i18n.out

`The first line of sample input.`
`The s4d l2e of s4e i3t.`
`Lone, lone, l4e...`
`S4e input last l4e!`
	`The first line of sample input.`
	`The s4d line of sample input.`
	`Lone, lone, l4e...`
	`Sample input last l4e!`

Problem J. Japanese Writing

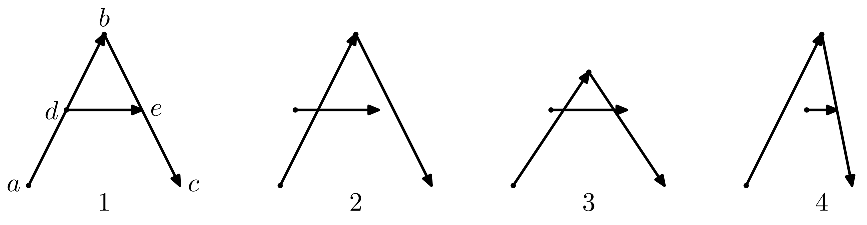
Input file:	`japanese.in`
Output file:	`japanese.out`

Michael had visited ACM ICPC World Finals 2007 in Tokyo, Japan and became fascinated with Japanese writing. He decided to study hieroglyphs, but in order to check his knowledge he needs a piece of software that can verify correctness of his writing. This program takes a description of a correct shape of the hieroglyph, several Michael's attempts to write it, and judges each attempt as correct or incorrect.

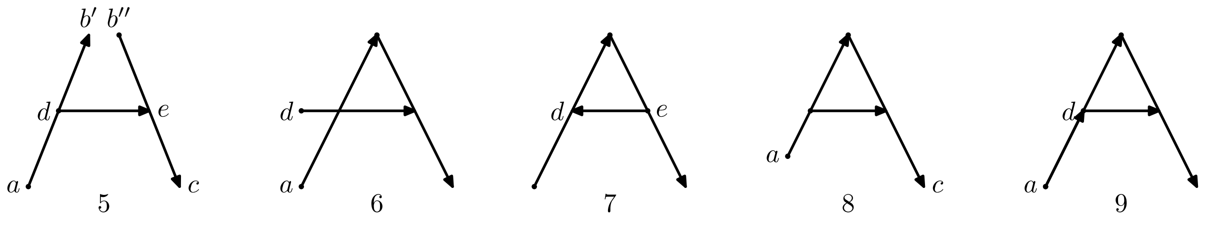
In this problem hieroglyphs are represented as a collection of strokes, each stroke being a straight line on a Cartesian plane. The order of strokes is irrelevant for the hieroglyph shape, but the direction of each stroke is relevant. There are eight relevant directions: straight right, upper-right, straight up, upper-left, straight left, lower-left, straight down, and lower-right.

Two writings are considered to represent the same shape if one-to-one correspondence can be established between the strokes and all the endpoints of the strokes, so that direction of strokes and relative positions of pairs of points are preserved. Preservation of relative positions is important for any pair of points, even if they are not connected with a stroke. However, relative positions are important only with respect to eight relevant directions described above.

For example, here is a hieroglyph similar to Latin letter A with 5 endpoints connected with 3 strokes and several other correct writings of the same shape:



Note, that intersections of strokes are not relevant. Here are several incorrect writings of the same shape:



These writings are not correct for the following reasons:

- Writing 5 has an extra point, so one-to-one correspondence between endpoints cannot be established.
- In writing 6 point d is straight up from point a but it should be to the upper-right of it.
- In writing 7 stroke $d - e$ goes in the wrong direction.
- In writing 8 point c is lower-right from point a but it should be straight to the right of it.
- Writing 9 has an extra $a - d$ stroke, so one-to-one correspondence between strokes cannot be established.

Input

The first line of the input file contains a single integer n ($2 \le n \le 20$) — the number of writings in the input file. It is followed by descriptions of n writings.

Each writing starts with a line with a single integer number m_i ($1 \le m_i \le 100$) — the number of strokes in i-th writing. It is followed by m_i lines that describe strokes for i-th writing. Each stroke is represented by a line with four integer numbers x^a_{ij}, y^a_{ij}, x^b_{ij}, and y^b_{ij} ($-1000 \le x^a_{ij}, y^a_{ij}, x^b_{ij}, y^b_{ij} \le 1000$) — coordinates of endpoints. x^a_{ij}, y^a_{ij} are coordinates of the beginning of j-th stroke and x^b_{ij}, y^b_{ij} are coordinates of the end of j-th stroke. The beginning and the end of the stroke are distinct points. Any two endpoints are connected by at most one stroke.

Output

Compare the shapes of writings from 2-nd to n-th with the shape of the first writing and write to the output file $n-1$ lines with the result of each comparison of a single line. Write `CORRECT` if the corresponding writing represents the same shape as the first one or `INCORRECT` otherwise.

Sample input and output

japanese.in	japanese.out

```
9
3
0 0 10 20
10 20 20 0
5 10 15 10
3
0 0 10 20
10 20 20 0
2 10 13 10
3
0 0 10 15
10 15 20 0
5 10 15 10
3
8 10 12 10
0 0 10 20
10 20 14 0
3
0 0 8 20
12 20 20 0
5 10 15 10
3
0 0 10 20
10 20 20 0
0 10 15 10
3
0 0 10 20
10 20 20 0
15 10 5 10
3
2 4 10 20
10 20 20 0
5 10 15 10
4
0 0 10 20
0 0 5 10
10 20 20 0
5 10 15 10
```
	```
	CORRECT
	CORRECT
	CORRECT
	INCORRECT
	INCORRECT
	INCORRECT
	INCORRECT
	INCORRECT
	```

Problem K. Kingdom Partitioning

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

The Kingdom of Qari was conquered, and now n other Kingdoms are dividing its territory among themselves. However, each Kingdom has its own opinion on which parts of Qari's land are better. For example, Napaj might want a large open space for settlement while Acirema is only interested in rich oilfields.

To formalize their claims each Kingdom has indicated an area that it considers worthy. An area indicated by each Kingdom is a union of non-overlapping circles. A Kingdom is content with Qari's partitioning if it receives at least $1/n$ of the area it has indicated as worthy.

Your task is to propose a partitioning that satisfies all n Kingdoms. In your partitioning an area given to each Kingdom must be a convex polygon with at most 1000 vertices. In this problem it is always possible to find such partitioning.

Input

The first line of the input file contains an integer number n ($1 \le n \le 30$) — the number of Kingdoms that are dividing Qari. Then follow n sections describing the areas that were indicated by each Kingdom as worthy.

The first line of each section contains an integer number m_i ($1 \le m_i \le 30$) — the number of circles indicated by i-th Kingdom. It is followed by m_i lines describing circles, one circle per line. A circle is described by three integer numbers x, y, and r ($-1000 \le x, y \le 1000$; $1 \le r \le 1000$) — the coordinates of its center and its radius correspondingly. Circles in one section do not intersect, but may touch each other.

Output

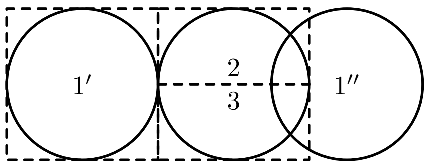
Write to the output file descriptions of n non-intersecting convex polygons — one for each Kingdom in the same order as in the input file. Polygons may touch each other (see sample output).

Each description shall start with a line that contains a single integer number k_i ($3 \le k_i \le 1000$) — the number of vertices in the polygon, followed by k_i lines with x and y coordinates of the vertices ($-3000 \le x, y \le 3000$). Coordinates must be precise up to 7 digits after decimal point. The vertices must be listed in either clockwise or counterclockwise direction. No three consecutive vertices are allowed to lie on the same straight line.

Sample input and output

kingdom.in	kingdom.out

3
2
0 0 2
7 0 2
1
4 0 2
1
4 0 2
	4
	2.0 2.0
	-2.0 2.0
	-2.0 -2.0
	2.0 -2.0
	4
	2.0 0.0
	6.0 0.0
	6.0 2.0
	2.0 2.0
	4
	2.0 0.0
	6.0 0.0
	6.0 -2.0
	2.0 -2.0



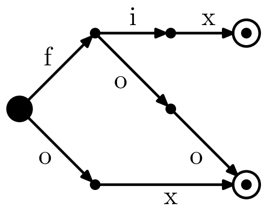
On the example picture above, the first Kingdom has indicated as worthy a union of circles 1′ and 1″. The second and the third Kingdoms have both indicated the circle marked with 2 and 3 as worthy. One of the possible partitionings is pictured with dashed lines — the first Kingdom will receive a square area around circle 1′ while the second and the third Kingdoms will receive rectangles 2 and 3, thus getting an equal share of their indicated area. With this partitioning each Kingdom gets half of the area it has indicated as worthy, which is more than one third required by the problem statement.

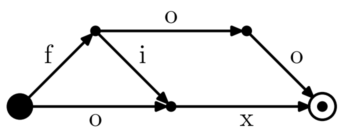
Problem L. Language Recognition

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

Deterministic Final-State Automaton (DFA) is a directed multigraph whose vertices are called *states* and edges are called *transitions*. Each DFA transition is labeled with a single letter. Moreover, for each state s and each letter l there is at most one transition that leaves s and is labeled with l. DFA has a single *starting* state and a subset of *final states*. DFA defines a language of all words that can be constructed by writing down the letters on a path from the starting state to some final state.

Given a language with a finite set of words it is always possible to construct a DFA that defines this language. The picture on the left shows such DFA for the language cosisting of three words: `fix`, `foo`, `ox`. However, this DFA has 7 states, which is not optimal. The DFA on the right defines the same language with just 5 states.





Your task is to find the minimum number of states in a DFA that defines the given language.

Input

The first line of the input file contains a single integer number n ($1 \le n \le 5\,000$) — the number of words in the language. It is followed by n lines with a word on each line. Each word consists of 1 to 30 lowercase Latin letters from "`a`" to "`z`". All words in the input file are different.

Output

Write to the output file a single integer number — the minimal number of states in a DFA that defines the language from the input file.

Sample input and output

`language.in`	`language.out`
3 fix foo ox	5
4 a ab ac ad	3