Problem A. Unique Attack

Input file:	`attack.in`
Output file:	`attack.out`
Time limit:	1 second
Memory limit:	64 megabytes

N supercomputers in the United States of Antarctica are connected into a network. A network has a simple topology: M different pairs of supercomputers are connected to each other by an optical fibre. All connections are two-way, that is, they can be used in both directions. Data can be transmitted from one computer to another either directly by a fibre, or using some intermediate computers.

A group of terrorists is planning to attack the network. Their goal is to separate two main computers of the network, so that there is no way to transmit data from one of them to another. For each fibre the terrorists have calculated the sum of money they need to destroy the fibre. Of course, they want to minimize the cost of the operation, so it is required that the total sum spent for destroying the fibres was minimal possible.

Now the leaders of the group wonder whether there is only one way to do the selected operation. That is, they want to know if there are no two different sets of fibre connections that can be destroyed, such that the main supercomputers cannot connect to each other after it and the cost of the operation is minimal possible.

Input

The first line of the input file contains N, M, A and B ($2 \le N \le 800$, $1 \le M \le 10000$, $1 \le A, B \le N$, $A \ne B$), specifying the number of supercomputers in the network, the number of fibre connections, and the numbers of the main supercomputers respectively.

Next M lines describe fibre connections. For each connection the numbers of the computers it connects are given and the cost of destroying this connection. It is guaranteed that all costs are non-negative integer numbers not exceeding 10^5, no two computers are directly connected by more than one fibre, no fibre connects a computer to itself and initially there is the way to transmit data from one main supercomputer to another.

Output

If there is only one way to perform the operation, output “`UNIQUE`”. In the other case output “`AMBIGUOUS`”.

Example

`attack.in`	`attack.out`

`4 4 1 2`
`1 2 1`
`2 4 2`
`1 3 2`
`3 4 1`
	`UNIQUE`

`4 4 1 2`
`1 2 1`
`2 4 1`
`1 3 2`
`3 4 1`
	`AMBIGUOUS`

Problem B. Burning Bridges

Input file:	`bridges.in`
Output file:	`bridges.out`
Time limit:	1 second
Memory limit:	64 megabytes

Ferry Kingdom is a nice little country located on N islands that are connected by M bridges. All bridges are very beautiful and are loved by everyone in the kingdom. Of course, the system of bridges is designed in such a way that one can get from any island to any other one.

But recently the great sorrow has come to the kingdom. Ferry Kingdom was conquered by the armies of the great warrior Jordan and he has decided to burn all the bridges that connected the islands. This was a very cruel decision, but the wizards of Jordan have advised him no to do so, because after that his own armies would not be able to get from one island to another. So Jordan decided to burn as many bridges as possible so that is was still possible for his armies to get from any island to any other one.

Now the poor people of Ferry Kingdom wonder what bridges will be burned. Of course, they cannot learn that, because the list of bridges to be burned is kept in great secret. However, one old man said that you can help them to find the set of bridges that certainly will not be burned.

So they came to you and asked for help. Can you do that?

Input

The first line of the input file contains N and M — the number of islands and bridges in Ferry Kingdom respectively ($2 \le N \le 10\,000$, $1 \le M \le 100\,000$). Next M lines contain two different integer numbers each and describe bridges. Note that there can be several bridges between a pair of islands.

Output

On the first line of the output file print K — the number of bridges that will certainly not be burned. On the second line print K integers — the numbers of these bridges. Bridges are numbered starting from one, as they are given in the input file.

Example

`bridges.in`	`bridges.out`

`6 7`
`1 2`
`2 3`
`2 4`
`5 4`
`1 3`
`4 5`
`3 6`
	`2`
	`3 7`

Problem C. Circles

Input file:	`circles.in`
Output file:	`circles.out`
Time limit:	1 second
Memory limit:	64 megabytes

Consider N different circles on the plane. They divide it to several parts, you have to find the number of these parts.

For the purpose of this problem the circle of radius r with center (x_0, y_0) is the set of points

$$C = \left\{(x, y) \,|\, (x - x_0)^2 + (y - y_0)^2 = r^2\right\}.$$

Input

The first line of the input file contains N — the number of circles ($1 \le N \le 50$). Next N lines contain three integer numbers x_0, y_0, and r each — the coordinates of the center and the radius of the circle.

All coordinates do not exceed 10^3 by their absolute value, all radii are positive and do not exceed 10^3.

No two circles coincide.

Output

First output K — the number of parts circles divide the plane to.

Due to floating point precision losses possible, do not consider parts with area not exceeding 10^{-10}.

Example

circles.in	circles.out

2
0 0 3
0 0 2
	3

Problem D. Linear Programming Dual

Input file:	`dual.in`
Output file:	`dual.out`
Time limit:	1 second
Memory limit:	64 megabytes

In this problem you have to find the dual to the linear programming problem given in general form.

Linear programming problem is set in the following way. For $i = 1, 2, \ldots, n$ let x_i be a variable. Some x_i may be required to be non-negative or non-positive. The goal is to minimize or maximize the sum $c_1x_1 + c_2x_2 + \ldots + c_nx_n$ under a set of restrictions.

Let $A = (a_{ij})$ be an $m \times n$ matrix. Denote the product $a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n$ as A_ix. Each restriction has the form $A_ix \ge b_i$, $A_ix \le b_i$ or $A_ix = b_i$.

Dual to this linear program is defined as the linear program with variables y_i for $i = 1, 2, \ldots, m$ (the number of dual variables is equal to the number of restrictions in the primal problem). If the primal problem was a minimization problem, the dual is a maximization one and vice versa. The objective function of the dual problem is $b_1y_1 + b_2y_2 + \ldots + b_my_m$.

Each dual variable is associated with the restriction of the primal problem. If the primal problem was a maximization one, variables, associated with $A_ix \le b_i$ restrictions must be non-negative and variables associated with $A_ix \ge b_i$ restrictions must be non-positive. In case the primal problem was a minimization one, the variables, associated with $A_ix \ge b_i$ restrictions must be non-negative and variables associated with $A_ix \le b_i$ restrictions must be non-positive. Variables associated with $A_ix = b_i$ restriction may be arbitary in either case.

The restrictions in the dual problem use matrix A^T (A transposed) instead of A. Restricitions have one of the form $A_i^Ty_i \le c_i$, $A_i^Ty_i \ge c_i$, or $A_i^Ty_i = c_i$. You must determine the type of the restriction using the idea that the dual to the dual problem is the primal problem again.

Given primal linear programming problem, output its dual.

Input

The first line of the input file contains n and m ($1 \le n, m \le 100$).

The second line of the input file contains the objective function of the given problem. The first word of the line is either "`min`" or "`max`", the objective function follows. All c_i are integer, x_i is given as "`x`*i*", multiplication sign is omitted, minus sign may replace the plus sign when needed to state that c_i is negative. Terms with $c_i = 0$ are omitted. If all c_i are zero, "`0`" is given as the objective function. Terms with $c_i = \pm 1$ are given without '`1`' character.

The third line contains the word "`with`".

Next n lines contain non-negativity and non-positivity conditions for the variables. If the variable may be arbitary, the word "`arbitary`" is used.

The next line contains the word "`under`".

Following m lines contain restrictions, all a_{ij} and b_i are integer, multiplication sign is omitted, terms with $a_{ij} = 0$ are omitted, if for some i all a_{ij} are zero, "`0`" is used as the left side of the equation. Terms with $a_{ij} = \pm 1$ are given without '`1`' character.

Input file contains no extra spaces.

Output

Output the dual to the problem given in the input file, in the same format. Remember, that the first line must contain the number of variables and restrictions in the *dual* problem, so that the output file produced is the valid input file for the problem with the exception that it has '`y`' instead of '`x`'.

You must list variables in all statements in increasing order of their numbers and must omit terms equal to zero. You must omit ±1 coefficient where needed, keeping only the sign. You must not print '+' if the following term has the negative coefficient.

Example

dual.in	dual.out
3 4 min 2x1-x3 with x1>=0 x2 arbitary x3<=0 under x1+x2+x3>=0 3x2=3 -x1-x2+100x3<=-4 0=0	4 3 max 3y2-4y3 with y1>=0 y2 arbitary y3<=0 y4 arbitary under y1-y3<=2 y1+3y2-y3=0 y1+100y3>=-1

Problem E. DVD

Input file:	`dvd.in`
Output file:	`dvd.out`
Time limit:	1 second
Memory limit:	64 megabytes

Johnie likes watching videos very much. Recently his old uncle Arnie has died and handed him down his collection of DVDs.

Johnie loved his uncle, so he must comply with his last will. Uncle Arnie was a possionate cinema adorer, so he wanted Johnie to get the full taste of XX-th century cinematograph. So he declared in his last will that Johnie must view the films he left him in chronological order. That is, if the film A was produced in year Y_A and the film B in year Y_B where $Y_A < Y_B$, Johnie must watch the film A before the film B. Films produced during one year may be watched in any order.

Johnie has already prepared to start watching the movie collection when a small problem has appeared: modern DVDs are protected with so called *regional protection*. That is, the world is divided into five parts, and each DVD can only be viewed in the region it is produced for.

Since DVD players are exported to different parts of the world, after the user buys the player, he is allowed to select the region he lives in. To correct occasional mistakes, the regional setting can be changed. However, to prevent changing region before watching each movie, the number of changes is limited to five.

Uncle Arnie loved movies so much, that he had five DVD players in his house, so he could view DVDs from all regions. However, Johnie is not so crazy about video yet, so he has only one DVD player. The collection of uncle Arnie contains many discs from various regions.

Help Johnie to select the set of films to watch and the order in which to do that. Initially his DVD player has no regional setting, so before watching the first film he must set its region. After that he can change the region for at most five times. Johnie wants to watch as many films as possible.

Input

The first line of the input file contains n — the number of films in the collection ($1 \le n \le 2000$). Next n lines describe films: the name of the film in double quotes is followed by its production year and the region of the DVD. Note that uncle Arnie never had several copies of one film in his collection and no two films have the same name. Year ranges from 1870 to 2004. No film name is longer than 100 characters.

Output

On the first line of the output file print m — the maximal number of films Johnie can watch. Next m lines must contain the names of the films in order Johnie must watch them.

Example

dvd.in	dvd.out

`10`
`"Gone with the Wind" 1960 1`
`"The World and the War" 1991 5`
`"Back to the Future" 1980 1`
`"The Wall" 1980 2`
`"Titanic" 1997 3`
`"Hell on Earth" 1980 3`
`"Princess" 1980 4`
`"Yesterday" 1980 5`
`"Shadows of the Triumph" 1984 3`
`"Help" 1965 2`
	`9`
	`"Gone with the Wind"`
	`"Help"`
	`"The Wall"`
	`"Princess"`
	`"Yesterday"`
	`"Back to the Future"`
	`"Hell on Earth"`
	`"Shadows of the Triumph"`
	`"Titanic"`

Problem F. Think Positive

Input file:	`positive.in`
Output file:	`positive.out`
Time limit:	1 second
Memory limit:	64 megabytes

It is well known, that the year on planet Eisiem has n days. Of course, some days are very good for people, while some others are just horrible. Long observations have shown for each day of the year whether this day is good for most people, or bad.

The new president of the Planet Federation wants all people to be happy. He knows that good emotions have a tendency to accumulate, just like bad ones do. The New Year however is a special event and all emotions accumulated by this moment just disappear. Therefore the president wants to change the calendar on Eisiem and choose the new first day of the year, so that the positive emotions would prevail the whole year.

More precisely, for all i from 1 to n let a_i be 1 if i-th day is good for most people and -1 if it is bad. Let s_{jk} be the sum of a_i for all days from the j-th day of the year to the k-th, that is:

$$s_{jk} = \begin{cases} \sum\limits_{i=j}^{k} a_i, \text{ if } k \ge j; \\ \sum\limits_{i=j}^{n} a_i + \sum\limits_{i=1}^{k} a_i, \text{ if } k < j. \end{cases}$$

President wants to find such j to order the j-th day to be the first day of the year, that s_{jk} is positive for all k from 1 to n. Since he wants several variants to choose from, he asks you to find all such j. Since he doesn't want to get too much information at once, first of all he wants to know the number of such j. That is exactly your task.

Input

The first line of the input file contains n — the number of days ($1 \le n \le 200\,000$). Next line contains n integer numbers — a_i.

Output

Output the number of different indices j, such that s_{jk} is positive for all k.

Example

`positive.in`	`positive.out`

`5`
`1 -1 1 -1 1`
	`1`

Problem G. Ranking

Input file:	`ranking.in`
Output file:	`ranking.out`
Time limit:	1 second
Memory limit:	64 megabytes

In this problem you are asked to implement the ranking system for a series of programming contests.

Each contest runs using standard ACM ICPC rules, each run is judged and either accepted, or rejected. For each solved problem the penalty time is calculated as the time of the first accepted run in minutes plus twenty minutes for each unsuccessful run before the first successful run. For problems that are not solved no penalty time is considered. The teams are first ranked by the number of solved problems, next by the penalty time. Each team gets the highest possible rank, that is, if two or more teams have the same number of solved problems and the same penalty times, they all share the same place, highest from the range they occupy. So, for example, there can be two teams sharing the first place, followed by the third place team, and so on.

The ranking of each team for the series of the contests is calculated as follows. Let P_{ij} be the number of problems solved by j-th team in the i-th contest. Let PM_i be the maximal number of problems solved by some team in the i-th contest. The *raw score* of the j-th team for this contest is $RS_{ij} = P_{ij}/PM_i$ if $PM_i > 0$ and $RS_{ij} = 0$ if $PM_i = 0$. Let K_i be the number of teams take part in the i-th contest. Calculate A_i and B_i, satisfying equations

$$\frac{A_i}{1 + B_i} = 2; \quad \frac{A_i}{K_i + B_i} = 1.$$

If the j-th team is ranked R_{ij} in the i-th contest, its *score* for this contest is

$$S_{ij} = RS_{ij}\frac{A_i}{R_{ij} + B_i}.$$

If the j-th team took part in $C_j > 0$ contests, its *total score* is

$$T_j = \frac{\sum\limits_i S_{ij}}{C_j},$$

in the other case its total score is $T_j = 0$.

Given the description of several contests, your task is for each team to calculate its total score.

Input

The first line of the input file contains N — the number of teams ($2 \le N \le 100$), next N lines contain team names, length of each name does not exceed 100 characters. Let teams be numbered as they are given in the input file, starting from one.

Next line contains M — the number of contests in a series ($1 \le M \le 20$). The descriptions of the contests follow. The first line of the contest descrption contains K_i — the number of teams that took part in this contest ($2 \le K_i \le N$) and the numbers of these teams. Next line contains PN_i — the number of problems in the contest ($1 \le PN_i \le 26$). Problems are identified using capital letters of the English alphabet, starting from 'A'. Next line contains RN_i — the number of runs in the contest ($0 \le RN_i \le 10\,000$). Next RN_i lines describe runs, each run description consists of four items, adjacent items are separated from each other by exactly one space. The items are: the number of the team, the letter of the problem, the time of the run in minutes and character '`+`' if the run is accepted, or '`-`' if it is rejected. Runs are given in the order they were made, in particular run times are non-decreasing. Times are positive integers that do not exceed 300.

Output

Output N lines — the teams in the non-increasing order of their total score. In case of equal total score, teams may be output in arbitary order. Print score aligned, so that there is exactly one space between the longest team name and the leftmost digit of the score, and all decimal points are in the same column. Print exactly four digits after the decimal point.

Example

ranking.in	ranking.out
4	MosCow SU 2.0000
MosCow SU	SPb IMHO 1.1667
ThreeThreads	ThreeThreads 1.1250
SPb IMHO	SPb FLY 0.0000
SPb FLY	
3	
3 1 2 3	
5	
10	
1 A 30 +	
1 B 80 +	
2 A 85 -	
2 C 90 +	
1 E 101 +	
3 A 130 -	
3 A 140 +	
1 D 150 +	
3 D 280 +	
3 E 299 +	
2 2 3	
2	
3	
2 A 160 +	
2 B 245 +	
3 A 280 +	
3 1 3 4	
4	
6	
1 A 50 +	
3 A 50 +	
1 A 155 -	
3 A 160 +	
1 B 180 -	
4 A 200 -	

Problem H. Driving Straight

Input file:	`straight.in`
Output file:	`straight.out`
Time limit:	1 second
Memory limit:	64 megabytes

The city where Peter lives has the form of the rectangle with M streets running from east to west and N streets running from north to south. Recently, because of preparations for the celebration of the city's 3141-st birthday, some street sectors has been closed for driving.

Peter lives in the house next to point $(1, 1)$ and works next to the point (N, M). He always drives from home to work by his car using the shortest possible path. Of course, he can't drive through closed sectors. Since there can be many shortest pathes between his house and his work, he may choose any.

But Peter doesn't like to turn (he is an inexperienced driver), so he wants to choose the path using the following algorithm: starting from the point $(1, 1)$ he drives either northwards, or eastwards (wherever there is the shortest path available, if there are both, he may choose any). Whenever he comes to the junction he must decide where to go. If there is only one direction he can drive to stay on the shortest path, he must choose that direction. In the other case he would like to choose the direction giving priority to driving forward, that is, if he can drive forward and still stay on some shortest path, he would go forward. If he can't drive forward to stay on the shortest path, he would choose any available direction.

Help Peter to create the path from his house to his work using the rules described.

Input

The first line of the input file contains integer numbers M and N ($2 \le M, N \le 400$).

Next $2M - 1$ lines contain $2N - 1$ characters each, representing the city map. House blocks are marked with spaces, junctions with '`+`', open sectors with '`-`' and '`|`', closed sectors with spaces. Peter's house is at the lower-left corner of the map, his work is at the upper-right corner.

Output

On the first line of the output file print the direction Peter should choose first, '`N`' or '`E`'. On the second line output the sequence of latin letters '`F`', '`L`' and '`R`' representing Peter's behaviour on junctions — go forward, turn left or right respectively. If there are several paths Peter can choose from, output any. You must output Peter's action on the junction even if he has no choice due to closed streets. It is guaranteed that there will always be the way for Peter to get from home to work.

Example

straight.in
```
4 4
+-+ +-+
| |   |
+ +-+-+
|   |
+-+-+-+
|     |
+-+-+-+
```

straight.out
```
N
RFLRL
```