Problem A
Contest Postmortem

If a chessboard position can be evaluated and assigned a numerical value, then it should be possible to do the same for a contest problem set. The following rules outline one possible scoring system:

A. Fairness:
One point if every problem has been solved by at least one team.

B. Inclusiveness:
Two points if more than 90% of the teams solved a minimum of two problems.

X. Challenge:
Two points if no team solved all the problems.

Your task is to write a program to score a problem set based on the above rules.

Input

Input consists of multiple cases. Each case starts with two integers on a separate line. The first integer **C** (**10** ≤ **C** ≤ **100**) represents the number of teams in the contest, and the second integer **P** (**8** ≤ **P** ≤ **20**) represents the number of problems in the set. The last case is followed by a line containing two zeros that indicates the end of the input data and should not be processed as a valid case. Each of the next **C** lines describes the performance of a single team. Each such line starts with the name of a team followed, after a blank space, by **P** integers. The **k***th* integer (**1** ≤ **k** ≤ **P**) has a value of one (1) to indicate that the team has solved the **k***th* problem, or zero (0) otherwise.

Output

For each contest, print the contest number (starting with 1, and using the format in the sample) followed by an integer indicating the calculated score.

Sample Input	Output for the Sample Input
10 8 Gladiators 1 1 1 1 1 1 1 1 Just4Pizza 1 1 1 1 1 0 1 0 The_greatests 1 1 1 0 1 0 0 0 2+1=us 1 0 1 0 0 1 0 1 we_are_1+2 0 0 1 1 0 1 1 1 random 1 0 0 0 1 1 1 1 cfjaszmubdfub 1 1 0 0 1 1 0 1 wbkdfevtmismxg 0 1 0 1 1 1 0 0 soxkukbmirk 0 1 1 0 0 1 0 0 axoqjkpwequsara 0 1 0 0 0 0 1 1 10 8 Gladiators 0 1 0 1 1 0 0 1 Just4Pizza 0 1 1 1 1 0 0 0 we_are_1+2 0 0 1 0 1 0 0 0 random 0 0 1 0 1 0 1 0 ugjzbdglfbktscq 0 1 0 0 0 0 0 1 vxxltjgrexz 0 0 0 1 0 0 1 0 xqapfogqfilqbta 0 1 1 0 0 0 1 0 mbgjlmcgmkkan 0 1 1 0 0 0 1 0 The_greatests 0 1 0 0 0 0 0 0 2+1=us 0 0 0 0 0 1 1 0 10 8 Gladiators 0 0 1 0 1 0 0 0 Just4Pizza 0 1 1 0 1 1 0 0 random 1 0 0 0 0 1 1 0 we_are_1+2 1 1 1 1 0 1 0 1 2+1=us 0 1 1 1 1 0 1 0 zumwuoezqqcmmc 1 0 1 1 0 0 0 0 fqabkrsrjg 0 1 0 1 1 1 0 1 pocdkprlpeva 1 0 1 0 0 1 1 0 The_greatests 0 0 0 0 0 0 0 1 nurtvuldyyrsa 1 1 1 1 0 1 1 1 10 8 Gladiators 1 1 1 1 1 1 1 1 Just4Pizza 0 0 0 1 1 1 0 0 The_greatests 0 0 0 0 0 0 0 0 we_are_1+2 1 1 1 0 1 0 1 1 random 0 0 1 1 0 0 1 0 ytypiowjhsok 0 1 1 0 0 1 0 0 gxvelxfbprutp 1 1 1 1 1 0 0 1 bdahyifafvrtzrc 0 0 1 1 1 1 1 1 2+1=us 1 0 1 0 0 1 1 1 koyzvguhyj 1 0 1 0 1 1 1 1 0 0	Contest 1: 3 Contest 2: 2 Contest 3: 3 Contest 4: 1

Problem B
and the winner is!

During junior chess tournaments, a player may be paired in matches with another player of a different age. However, at the end of the tournament, players with the best score for each age (where age is calculated on the day of the tournament) are recognized separately. In the case of a tie, all players attaining the highest score are recognized. For the sake of clarity, a birthday is defined as the day that a person turns a year older. For example, a person born on the 2^{nd} June 1996 is five (5) years old on the 1^{st} June 2002, is six (6) years old on the 2^{nd} June 2002 and will be considered in the age category of six (6) in a tournament held on the 2^{nd} June 2002. Note that a person born on the 29^{th} February becomes a year older on the 29^{th} February in leap years and on the 1^{st} March in non-leap years.

Your task is to write a program to identify the winners, and co-winners, amongst those participating in a junior chess tournament.

Input

The input starts with an integer **N**, $1 \leq \mathbf{N} \leq 200$, on a separate line that represents the number of contests to be processed. The description of each contest starts with the contest date. The date is given as three entries on a separate line in the form: <day> <month> <year>, where<day> is an integer whose value is inclusive of one (1) and the number of days in the month, <month> is a string with no white spaces and <year> is an integer. Entries are separated by a blank space.

The following line consists of an integer **M**, $1 \leq \mathbf{M} \leq 1000$, which represents the number of contestants followed by **M** lines. Each of the **M** lines contains information about a single contestant. Contestant information is given as three entries on a separate line in the form:

<LastName> <FirstName> : date of birth : score

Names are separated by a single space and there are no white spaces in a name, date of birth uses the same format as the contest date and score is an integer inclusive of 200 and 2000. Entries are separated by a colon (:) with a white space on each side.

Output

For each contest the output starts with a line that contains the contest number (starting with 1) followed by a colon (:), followed by a space and then followed by the contest date in the same format as that used in the input. The next line contains the profound statement All participants are winners with.

Each of the following lines identifies the winner in an age category in the following form:

<LastName> < FirstName> is best under <age>.

There are no white spaces in a name and age is an integer that represents the age categories in years. In the case of a tie, a separate line is added in the form:

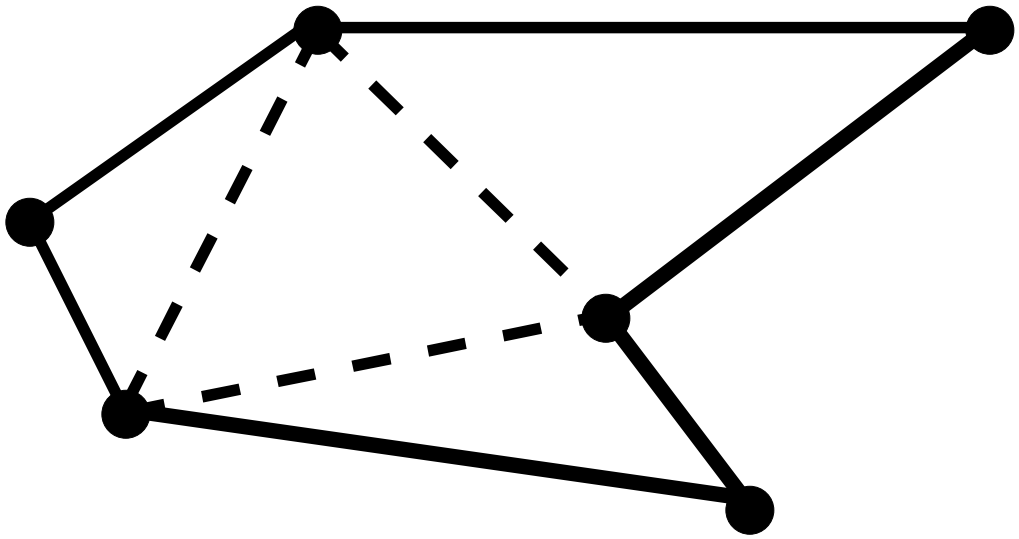
and also <LastName> < FirstName>.

The age categories are listed in increasing order.

Sample Input	Output for the Sample Input
2 24 february 2017 10 dnuyixad uft : 28 october 2009 : 253 gzadkv eyht : 1 october 2004 : 1103 uxnlbc nu : 1 july 2007 : 700 ekrasl rtf : 18 september 2005 : 349 kjbu pgc : 18 november 2001 : 299 gtysh veuvzb : 1 april 2003 : 856 ldywn klyj : 6 november 1999 : 268 hmincdt mym : 6 july 2009 : 705 pcik nomm : 11 march 2004 : 755 dmjldfx rzlilm : 3 november 2000 : 901 12 june 2010 10 tydruqdh jfypo : 30 june 1996 : 1183 rhkefcr cz : 16 july 2001 : 981 bdpoq ltgrh : 26 october 1997 : 823 kcobbp rctwd : 23 december 1998 : 280 ssbdsoty auhk : 26 november 2001 : 511 vglu bn : 4 august 1997 : 823 cnenbiyu dia : 4 august 1997 : 500 hodjuadh pbl : 13 october 1995 : 835 rtzrtepz zlpk : 7 july 1995 : 493 qlyczjxf ppmvx : 15 april 1994 : 226	Contest 1: 24 february 2017 All participants are winners with hmincdt mym is best under 8 uxnlbc nu is best under 10 ekrasl rtf is best under 12 gzadkv eyht is best under 13 gtysh veuvzb is best under 14 kjbu pgc is best under 16 dmjldfx rzlilm is best under 17 ldywn klyj is best under 18 Contest 2: 12 june 2010 All participants are winners with rhkefcr cz is best under 9 kcobbp rctwd is best under 12 bdpoq ltgrh is best under 13 and also vglu bn tydruqdh jfypo is best under 14 hodjuadh pbl is best under 15 qlyczjxf ppmvx is best under 17

Problem C
Compact Triangulation

A simple polygon of size **M**, $3 \leq$ **M**, is a planar piecewise-linear closed curve of M distinct vertices that does not intersect itself. A chord is a line segment between two non-adjacent vertices of the polygon that lies entirely inside the polygon. In other words, the endpoints of the chord are the only points of the chord that touch the boundary of the polygon. A triangulation of the polygon is any choice of **M** −3 chords, such that the polygon is divided into triangles. In a triangulation, no two of the chosen chords intersect each other, except at endpoints.



An example of a simple polygon (border shown by solid line)
and one possible triangulation (with chords shown by dashed lines)

Constructing an arbitrary triangulation is fairly easy, but what is needed is to construct a triangulation whose largest area triangle is the smallest amongst all possible triangulations of a given simple polygon. Your task is to write a program to identify such a triangulation and produce the area of its largest triangle.

Input

The input consists of a series of triangulation requests with the first line being a single positive integer **N**, $0 < \mathbf{N} \leq 20$, that represents the number of polygons to follow. The description of each simple polygon starts with a line containing one positive integer **M**, $\mathbf{2} \leq \mathbf{M} \leq 49$, which represents the number of vertices of the polygon. Each of the following **M** lines contains a vertex of the polygon in order of their appearance along the border, going clockwise or counterclockwise, starting at an arbitrary vertex. Each vertex is described by a pair of integers x and y, $0 \leq x \leq 10\,000$ and $0 \leq y \leq 10\,000$, which represent its x- and y-coordinates.

Output

For each polygon, output one line that contains the area of the largest triangle in the identified triangulation. The area should be presented with one fractional decimal digit, using standard rounding rules. For example, 9.0x is truncated to 9.0 for all values of x in the range of one (1) to four (4), and 9.0y is rounded up to 9.1 for all values of y in the range of five (5) to nine (9).

Sample Input	Output for the Sample Input
1 6 7 0 6 2 9 5 3 5 0 3 1 1	9.0

Problem D
Blue Moon

In the land of ACMania the calendar is somewhat different to the Gregorian calendar that we are familiar with and use. The ACManian Calendar does not have 12 months and the months have an irregular number of days in them. They do, however still have a moon and it has a regular cycle with full moons being exactly the same number of days apart from each other. When the moon is full for the second time in the month, they call that a Blue Moon. Whenever there is a Blue Moon, the ACManians have a Blue Moon Festival that sometimes incorporates a programming competition.

Because of the irregularity in the calendar's cycle, the ACManians have trouble knowing the date of the next Blue Moon Festival. They have approached you to write a program that will calculate the date of the next festival.

Input

The input consists of a series of scenarios for calculating the date of the next Blue Moon Festival. The first line in each scenario consists of a positive integer, ***M***, which is the number of months in the year ($1 \leq \boldsymbol{M} \leq 1000$). The next line contains ***M*** positive integers that represent the number of days in each month of the calendar. Each of the ***M*** integers has a value inclusive of one (1) and five hundred (500). Consecutive integers are separated by a single space. The next line consists of a single positive integer that represents the number of days in the moon's cycle ($1 \leq \boldsymbol{P} \leq 1000$). The final line in the scenario gives the date of the last known full moon in *d/m/y* format where *d*, *m* and *y* are all positive integers. The date given will be a valid date based on the number of days in each month of the calendar. The *d* value will be inclusive of one (1) and the number of days in the month, and the *m* value will be inclusive of one (1) and the number of months in the calendar. A scenario with ***M*** equals zero (0) terminates the input data.

Output

Output consists of one line for each scenario. It will be in one of the following two formats:

The next Blue Moon Festival will be held on *d/m/y*.

There can never be a Blue Moon Festival.

Sample Input	Output for the Sample Input
10 20 20 20 20 25 20 20 25 20 25 21 4/2/2009 5 2 3 4 5 6 7 2/3/45 0	The next Blue Moon Festival will be held on 23/10/2009. There can never be a Blue Moon Festival.

Problem E
Rewards for Math!

My high school mathematics teacher was fond of challenging us to find as many distinct solutions of a given polynomial equation. There was a reward for the first pupil to report a correct solution, and a special reward for the pupil who declares that the last value has been reported. A polynomial of degree M has **M** solutions. Most pupils worked hard to quickly find any solution for the polynomial equation. However the clever ones aimed for solutions with multiplicity larger than one and thus got a shot at the special reward.

Your task is to write a program to check whether a given polynomial of degree **M** has **M** distinct, real or complex, solutions. The average students will greatly appreciate your help.

Input

The input starts with an integer **N** ($1 \le \mathbf{N} \le 100$) on a separate line that represents the number of equations to be checked. The description of each equation is given on a single line as a series of integers, which are separated by single blank spaces. The first integer **M** ($0 \le \mathbf{M} \le 10$) on each line represents the degree of the polynomial, followed by the **M+1** coefficients $a_0, a_1, \ldots a_M$ ($-30 \le a_i \le 30$, $a_0 \neq 0$) to form the equation $\sum_{i=0 \,..\, M} (a_i x^{M-i}) = 0$.

Output

Output consists of a single line for each equation. It will be in one of the following two forms:

Yes!	when the equation has **M** distinct solutions.
No!	when the equation has less than **M** distinct solutions.

Sample Input	Output for the Sample Input
4 2 1 1 1 2 1 2 1 4 1 2 1 2 1 4 1 2 2 2 1	Yes! No! Yes! No!

acm International Collegiate Programming Contest 2009 IBM event sponsor

South Pacific Region

Problem F
Easy does it!

The motivation for this problem is a directionally impaired friend, who doesn't care about the shortest or quickest routes -- instead, easier is better. Your task is to read the map information from a database of the entire country and produce the route with the smallest number of roads to use for a trip between an origin city and a destination city. You are guaranteed that there will be a path between origin and destination for any query asked.

Input

The input consists of multiple cases. The input for each case consists of three sections: the first lists the cities, the second lists the roads, and the third lists the queries. Each case starts with an integer **C** ($1 \leq \mathbf{C} \leq 200$) that represents the number of cities followed by **C** lines each containing the name of a city. There is no white space in a city's name.

The next line consists of an integer **R** ($1 \leq \mathbf{R} \leq 100$) that represents the number of roads followed by **R** lines each describing a road. Each road description has the following form:
<RoadName> <CityA> <ABDistance> <CityB> [<BCDistance> <CityC> [...]]
There is no white space in a road's name. Roads may pass through any number of cities. The cities appear in the order the road passes through them, and no road passes through the same city more than once. Roads are bidirectional. The distance (an integer number) it takes to follow a road between each pair of cities is the distance listed between these two names.

The next line consists of an integer **Q** ($1 \leq \mathbf{Q} \leq 100$) that represents the number of queries followed by **Q** lines each describing a query. Each query description has the following form: <origin> <destination>, where <origin> and <destination> are the names of cities.

A case with **C** equals zero (0) terminates the input data, and should not be processed.

Output

For each query, the output consists of a single line of the following form:
Number of roads needed from <origin> to <destination> is <number>.
<origin> and <destination> are the names of cities, and <number> is the smallest number of roads needed for the trip.

Sample Input	Output for the Sample Input
5 Adelaide Melbourne Sydney WestSydney Brisbane 4 OR Adelaide 300 Melbourne HH Melbourne 850 WestSydney 105 Sydney M7 WestSydney 1130 Brisbane BushTrack Adelaide 2448 Brisbane 1 Adelaide Brisbane 0	Number of roads needed from Adelaide to Brisbane is 1.

Problem G
Trip Verification

A travel agency that specializes in the sale of theme trips is currently planning a few theme trips for the coming holiday season. The agency wants to use regional airlines for traveling, to take advantage of their cheap prices, without affecting the following important features of their trips:

1. The order of cities in a trip cannot be altered.
2. Each city in a planned trip can be visited once and only once.

For example, a theme trip of "Recent Democracies" that consists of the following sequence of cities: Cairo, Kabul, Baghdad, and Benghazi may be altered to Cairo, Damascus, Nairobi, Kabul, Baghdad, Damascus and Benghazi. The altered trip may contain multiple visits to other cities in transit (Damascus in the above example) to take advantage of cheap flights without violating the above two important features.

Your task is to write a program to verify that the agency's trips can be planned using the available cheap regional airlines.

Input

The input consists of a series of trips whose plan is to be verified. The first line in each trip consists of a positive integer, M, $2 \leq M \leq 50$, which is the number of cities in the planned trips. The next line contains M strings that represent the unique names of cities to be visited in the given order. Each pair of city names is separated by a single blank space, and names of cities do not contain white spaces. The next line consists of a single positive integer P, $1 \leq P \leq 300$, which is the number of available flight segments followed by P lines each describing a flight segment. Each flight segment is a one-way flight whose description has the following form: <CityA> <CityB>. The pair of city names is separated by a single blank space, and names of cities do not contain white spaces.

A tour with ***M*** equals zero (0) terminates the input data, and should not be processed.

Output

Output consists of one line for each tour. It will be in one of the following two formats:

The tour can be planned.
The tour cannot be planned with the given flight segments.

Sample Input	Output for the Sample Input
4 Cairo Kabul Baghdad Benghazi 6 Cairo Damascus Cairo Benghazi Kabul Baghdad Baghdad Damascus Kabul Moscow Kabul Damascus 0	The tour cannot be planned with the given flight segments.

Problem H
Mentoring Assignment

The ACM, which is a charitable society for men, runs a program to support troubled adolescents via the help of mature volunteers. The program aims to pair an adolescent with a volunteer to provide the adolescent with the best possible mentoring and to provide the volunteer with a rewarding experience. Towards that goal, each volunteer and each adolescent is required to perform a personality-profiling test. A test reports a score for each of the **N** different personality traits. The scores are inclusive of one to a hundred. The report has the following form: PersonCategory <name> s1 s2 s3 ... sN, where PersonCategory is either Adolescent or Volunteer, and each si ($1 \le i \le$ **N**) is an integer value in the range of one (1) to one hundred (100), inclusive.

The ACM decided to assign weights to indicate the importance of each trait. The weights (*w1, w2, … wN*) used for adolescents are different from those weights (*v1, v2, … vN*) used for volunteers. The function

$$f_1(A, V) = (w1\,(A_{S1} - V_{S1})^2 + w2\,(A_{S2} - V_{S2})^2 + ... + wN\,(A_{SN} - V_{SN})^2)$$

is used to assign a numerical value to the quality of mentoring an adolescent A receives from volunteer V, and the function

$$f_2(A, V) = (v1\,(A_{S1} - V_{S1})^2 + v2\,(A_{S2} - V_{S2})^2 + ... + vN\,(A_{SN} - V_{SN})^2)$$

is used to assign a numerical value to the quality of experience volunteer V gets from mentoring adolescent A. A smaller value of f_1 indicates a higher quality of adolescent mentoring and a smaller value of f_2 indicates a higher quality of volunteer experience.

Assuming the names in each category are unique, your task is to write a program to prescribe a pairing of adolescents with volunteers such that no pair would be a better fit with each other than the pairing that your program prescribed for them. That is, no pair of an adolescent ***X*** and a volunteer ***Y*** has both values of $f_1(X, Y)$ and $f_2(X, Y)$ smaller than the values prescribed by your program for both of them.

Input

Input consists of multiple situations. Each situation starts with two integers on a separate line. The first integer **N** (**1** ≤ **N** ≤ **100**) represents the number of personality traits to be used, and the second integer **P** (**1** ≤ **P** ≤ **1000**) represents the number of adolescents and also the number of volunteers. The last situation is followed by a line containing two zeros that indicates the end of input data and should not be processed as a valid situation.

The second line contains **N** integers that describe the weights to be used for adolescents. Consecutive integers are separated by a single blank space, and each integer has a value of one (1) and ten (10) inclusive.

The third line contains **N** integers that describe the weights to be used for volunteers. Consecutive integers are separated by a single blank space, and each integer has a value of one (1) and ten (10) inclusive.

The following 2**P** lines describe the reports of 2**P** personality tests. Each such line starts with the PersonCategory followed, after a blank space, by a string with no white spaces that represents the name followed, after a blank space, by **N** integers. The **k***th* integer (**1** ≤ **k** ≤ **N**) has a value in the range of one (1) to one hundred (100) that represents the score for the **k***th* trait. The scores are separated by a blank space.

Output

For each situation, print the situation number (starting with 1, and using the format in the sample) on a separate line. The next **P** lines list the pairing of adolescents and volunteers (using the format in the sample) such that the adolescent names are listed in increasing lexicographic order.

Sample Input	Output for the Sample Input
2 3 1 1 1 1 Volunteer Smith 20 10 Adolescent Paul 20 30 Adolescent Peter 30 30 Volunteer Kevin 20 40 Volunteer Peter 50 10 Adolescent John 10 20 0 0	Situation 1: Adolescent John Volunteer Smith Adolescent Paul Volunteer Kevin Adolescent Peter Volunteer Peter

Problem I
May the Right-Force be with you

In a sensational scene in the 8th installment of the *Voldemort* book series, the robot *Weighd* was attacked and its rotating top and wings were severely damaged. *Weighd* cannot rotate its famous top and can only fly forward or make a right-turn. However, the brave *Weighd* must still navigate a magically created series of mazes to deliver its precious message to *Ttoper*.

Weighd enters each maze at the top-left square of the maze, takes one unit of time to travel between adjacent squares in the magic maze (forward or right), and cannot stay in one square to rotate itself, and thus change direction, for fear of evil jinx. Your task is to write a program to calculate the smallest number of squares that *Weighd* must travel to reach a certain square in each maze. Do not you worry about how? *Weighd* will feel its correct way through the force. Examples are:

start	f	f	f	f	r							
					f							
				r	f	f	f	f	f	f	f	r
				r	r							**X**

"f" means forward ,"r" means a right-turn and "?" means no where to go.
Note that the green square is crossed twice by Weighd.

start	f	f	f	f	r							
					f							
?	f	f	f	f	r/f							
					f							**X**
					?							

Weighd can reach the location marked "**X**" of the top maze in "19" steps, but it cannot reach the marked location in the bottom maze at all.

Input

Input consists of multiple mazes. Each maze description starts with two integers, on a separate line, that represent its dimensions. The first integer **N** ($1 \leq \mathbf{N} \leq 1000$) represents the number of rows and the second **M** ($1 \leq \mathbf{M} \leq 1000$) represents the number of columns. The last maze is followed by a line containing two zeros that indicate the end of the input data and should not be processed as a valid situation.

The second line contains two integers, C ($1 \leq C \leq N$) and R ($1 \leq R \leq M$), that represent the column number and the row number of the square where *Weighd* must reach in the maze. Consecutive integers are separated by a blank space.

Each of the following **N** lines contains a sequence of 0*s* and 1*s*. The sequence is M characters long. The sequence does not contains blank spaces. A value one (1) represents a wall in the maze (that is, a square into which *Weighd* cannot fly).

Output

Output consists of one line for each maze. It will be in one of the following two formats:

1. an integer that represents the number of steps to be taken, inside the maze, by *Weighd* to reach its destination.
2. The string "NOOO!", if no path can be found.

Sample Input	Output for the Sample Inpu
6 13 13 4 0000001111000 0111101111000 0000000000000 0000001111110 0000001111110 0000000000000 6 13 13 4 0000001111000 1111101111000 0000000000000 0001101111110 0001101111110 0001100000000 0 0	19 NOOO!