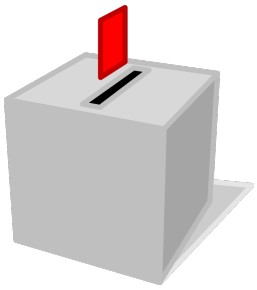
Problem A
Popular Vote
Problem ID: vote
Time Limit: 2 seconds



In an election with more than two candidates, it is often the case that the winner (the candidate receiving the most votes) receives less than the majority of the votes. Given the results of an election, can you determine the winner, and whether the winner received more than half of the votes?

Input

The first line of input contains a single positive integer $T \leq 500$ indicating the number of test cases. The first line of each test case also contains a single positive integer n indicating the number of candidates in the election. This is followed by n lines, with the ith line containing a single nonnegative integer indicating the number of votes candidate i received.

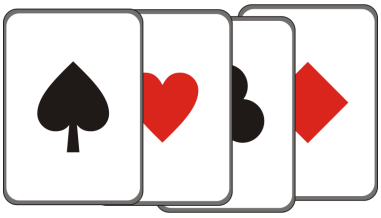
There are at least 2 and no more than 10 candidates in each case, and each candidate will not receive more than $50\,000$ votes. There will be at least one vote cast in each election.

Output

Provide a line of output for each test case. If the winner receives more than half of the votes, print the phrase `majority winner` followed by the candidate number of the winner. If the winner does not receive more than half of the votes, print the phrase `minority winner` followed by the candidate number of the winner. If a winner cannot be determined because no single candidate has more vote than others, print the phrase `no winner`. The candidate numbers in each case are $1, 2, \ldots, n$.

Sample Input	Sample Output
4 3 10 21 10 3 20 10 10 3 10 10 10 4 15 15 15 45	majority winner 2 minority winner 1 no winner minority winner 4

Problem B
Flipping Cards
Problem ID: flippingcards
Time Limit: 5 seconds



Mike and his young daughter Jesse are playing a new card game meant for kids. The rules are quite simple, each player is dealt a hand of cards. Each card has one picture on each side. They take turns playing cards and the first one to run out of cards is the winner.

A player's turn consists of picking a subset of cards from their hand and laying them on the table. The only rule is that the cards must be placed on the table such that no two cards are showing the same picture.

Mike thought this was a very appropriate game to play with his kid because of the simple rules. Mike also liked this game because finding the best strategy is an algorithmically interesting challenge!

Help Mike determine if he can play his entire hand on his first round.

Input

The first line of the input contains a single positive integer T ($T \leq 10$) indicating the number of test cases. Each test case begins with a single integer n denoting the number of cards in Mike's hand. Here $1 \leq n \leq 50\,000$. Following this are n lines, each describing a card in Mike's hand.

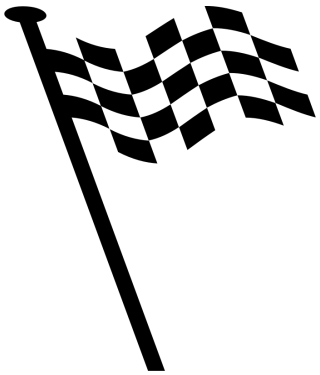
The pictures on the cards are represented by integers. The ith card is given by two integers p_i, q_i where $1 \leq p_i, q_i \leq 2n$.

Output

For each test case you should output a single line with the word `possible` if it is possible for Mike to play his entire hand in one turn, or `impossible` if Mike cannot play his entire hand in one turn.

Sample Input	Sample Output
3 3 1 2 1 3 2 3 3 1 2 1 2 1 2 1 1 1	possible impossible possible

Problem C
Amazing Race
Problem ID: race
Time Limit: 13 seconds



A scavenger hunt is being organized for programming contest participants. In addition to the starting and ending locations of the race, there are n ($n \leq 20$) other locations for competitors to travel to. At each location i ($1 \leq i \leq n$), there is a task that must be performed to earn p_i points. The task at each location takes t_i minutes to complete. However, each task can only be performed once, so a competitor may not travel to the same location more than once. The competitor cannot return to the starting location after the race begins, and the race finishes as soon as the ending location is reached.

The scavenger hunt must be completed within T minutes. That is, the time between leaving the starting location and arriving at the ending location must be no more than T minutes. In addition, some tasks have a specific deadline d_i, meaning that the task must be completed within d_i minutes since leaving the starting location. Again, note that if a competitor arrives at location i, the task at location i must be performed. If the competitor were to arrive at the location too late and would not finish the task at that location by the deadline, then the competitor would not be allowed to travel to the location at all.

What is the maximum total number of points that can be obtained from the tasks?

Input

The input consists of one case. The first line of input contains two positive integers n and T ($T \leq 1440$). Each of the next n lines contains three integers p_i ($1 \leq p_i \leq 100$), t_i ($1 \leq t_i \leq 1440$), and d_i ($-1 \leq d_i \leq 1440$). If $d_i = -1$ then there is no deadline for task i. Finally, the last $n + 2$ lines each contains $n + 2$ nonnegative integers. The entry in the ith row and jth column is the number of minutes (≤ 1440) it takes to travel from location i to location j. The indices of the starting and ending locations are $n + 1$ and $n + 2$, respectively.

It is guaranteed that the time to travel from a location to itself is 0, but the time to travel between two locations in different directions may not be the same (e.g. uphill instead of downhill).

Output

Print the maximum total number of points that can be obtained on the first line. In the second line, print a set of indices of the tasks that need to be performed to achieve this maximum. The indices should be separated by a single space. If there are multiple sets of tasks that can achieve the maximum, print the one that is lexicographically smallest. That is, if two sets of tasks achieve the same maximum, the index of the first task in the set should be as small as possible. If there is a tie, the index of the second task in the set should be as small as possible, and so on.

If the maximum number of points that can be obtained is 0, output a blank line for the indices of the tasks

to be performed.

Sample Input

```
3 352
93 82 444
92 76 436
99 62 -1
0 70 66 71 97
76 0 87 66 74
62 90 0 60 94
60 68 68 0 69
83 78 83 73 0
```

Sample Output

```
99
3
```

Sample Input

```
5 696
96 88 532
99 70 519
96 66 637
90 92 592
95 94 -1
0 67 80 81 60 83 61
72 0 99 68 85 93 82
100 91 0 88 99 70 68
69 65 77 0 65 68 75
63 65 91 96 0 92 100
65 76 85 62 89 0 75
93 83 74 65 88 84 0
```

Sample Output

```
386
1 2 3 5
```

Problem D
Scaling Recipes
Problem ID: recipes
Time Limit: 8 seconds

A recipe is a list of ingredients and a set of instructions to prepare a dish. It is often written for a particular number of portions. If you have a recipe for 4 portions and you want to make 6 portions, it turns out that simply multiplying the amounts for each ingredient by 1.5 is often wrong! The reason is that the original recipe may have been rounded to the nearest teaspoon, gram, etc., and the rounding errors magnify when a recipe is scaled.

Some recipes are specifically written to ease the task of scaling. These recipes are developed using "Baker's percentages." Each ingredient is listed not only by weight (in grams), but also as a percentage relative to the "main ingredient." The main ingredient will always have a 100% Baker's percentage. Note that the sum of the Baker's percentages from all ingredients is greater than 100%, and that the Baker's percentages of some ingredients may exceed 100%.

Table 1: Example Recipe

Ingredient	Weight (g)	Percentage (%)
Olive Oil	50.9	11.2
Garlic	12.0	2.7
Beef	453.6	100.0
Onions	1134.0	250.0
Raisins	82.5	18.2
Bouillon	10.0	2.2

To scale a recipe:

1. determine the scaling factor by dividing the number of desired portions by the number of portions for which the recipe is written;
2. multiply the weight of the main ingredient with a 100% Baker's percentage by the scaling factor. This is the scaled weight of the main ingredient;
3. calculate the scaled weight of every other ingredient by multiplying its Baker's percentage by the scaled weight of the main ingredient.

Input

The first line of input specifies a positive integer $T \leq 1000$, consisting of the cases to follow. Each case starts with a line with three integers R, P, and D: $1 \leq R \leq 20$ is the number of ingredients, $1 \leq P \leq 12$ is the number of portions for which the recipe is written, and $1 \leq D \leq 1000$ is the number of desired portions. Each of the next R lines is of the form

```
<name>   <weight>   <percentage>
```

where `<name>` is the name of the ingredient (an alphabetic string of up to 20 characters with no embedded spaces), `<weight>` is the weight in grams for that ingredient, and `<percentage>` is its Baker's percentage. Both `<weight>` and `<percentage>` are floating-point numbers with exactly one digit after the decimal point. Each recipe will only have one ingredient with a Baker's percentage of 100%.

Output

For each case, print `Recipe #` followed by a space and the appropriate case number (see sample output below). This is followed by the list of ingredients and their scaled weights in grams. The name of the ingredient and its weight should be separated by a single space. Each ingredient is listed on its own line, in the same order as in the input. After each case, print a line of 40 dashes ('-'). Answers within 0.1g of the correct result are acceptable.

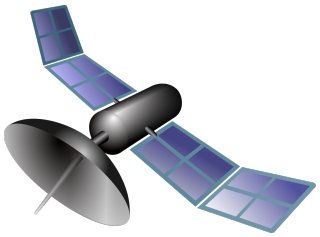
Sample Input

```
2
6 4 20
oliveoil 50.9 11.2
garlic 12.0 2.7
beef 453.6 100.0
onions 1134.0 250.0
raisins 82.5 18.2
bouillon 10.0 2.2
4 5 8
Milk 265.0 93.0
SodiumCitrate 11.0 4.0
WhiteCheddar 285.0 100.0
DryMacaroni 240.0 84.0
```

Sample Output

```
Recipe # 1
oliveoil 254.0
garlic 61.2
beef 2268.0
onions 5670.0
raisins 412.8
bouillon 49.9
----------------------------------------
Recipe # 2
Milk 424.1
SodiumCitrate 18.2
WhiteCheddar 456.0
DryMacaroni 383.0
----------------------------------------
```

Problem E
Space Junk
Problem ID: junk
Time Limit: 4 seconds



According to NASA's web page, there are more than $500\,000$ pieces of "space junk" that are tracked. Care must be taken in mission planning so satellites and other spacecrafts do not collide with these pieces of space junk.

For this problem, we will consider the simplified case in which both the spacecraft and the space junk can be modelled as spheres that are travelling in a straight line. Given the current locations of the two spheres as well as their velocities, when would they collide in the future, if ever?

Input

The first line of input contains a single positive integer $T \leq 500$ indicating the number of test cases. Each test case is specified by two lines. The first line specifies the sphere representing the spacecraft, while the second line specifies the sphere representing the space junk. Each sphere is specified by the seven integers $x, y, z, r, v_x, v_y, v_z$. The center of the sphere is currently located at (x, y, z), the radius of the sphere is r, and the sphere is travelling along the direction specified by the vector (v_x, v_y, v_z). If the vector is $(0, 0, 0)$, the sphere is stationary.

The absolute value of all integers are at most 100, and r is positive. All coordinates and radius are measured in meters, and the velocities are measured in meters/second.

You may assume that the two spheres do not touch each other initially.

Output

For each test case, output a line containing the time (in seconds) at which the spacecraft first collides with the space junk. If they never collide, print `No collision` instead. Answers within 0.01 of the correct result are acceptable.

Sample Input	Sample Output
3 10 3 -10 5 -9 3 -8 2 0 0 6 -4 3 -10 -7 5 0 3 -1 0 3 10 7 -6 6 -2 0 4 -4 -1 0 3 -1 -5 -6 2 1 8 6 4 0 -1	0.492 8.628 No collision

This page is intentionally left blank.

Problem F
A Classy Problem
Problem ID: classy
Time Limit: 6 seconds

In his memoir "So, Anyway", comedian John Cleese writes of the class difference between his father (who was "middle-middle-middle-lower-middle class" and his mother (who was "upper-upper-lower-middle class"). These fine distinctions between classes tend to confuse North American readers, so you are to write a program to sort a group of people by their classes to show their true place in the social class hierarchy.

For this problem, there are three main classes: upper, middle, and lower. Obviously, the highest is upper and the lowest is lower. But there can be distinctions within a class, so upper-upper is a higher class than middle-upper, which is higher than lower-upper. However, all of the upper classes (upper-upper, middle-upper, and lower-upper) are higher than any of the middle classes.

Within a class like middle-upper, there can be further distinctions as well, leading to classes like lower-middle-upper-middle-upper. When comparing classes, once you have reached the lowest level of detail, you should assume that all further classes are the same as the middle level of the previous level of detail. So upper class and middle-upper class are equivalent, as are middle-middle-lower-middle and lower-middle.

Input

The first line of input contains a single positive integer T ($T \leq 500$) indicating the number of cases to follow. Each case starts with a positive integer n ($n \leq 100$) on a line indicating the number of people to consider. Each of the next n lines contains the name of a person followed by a colon and a space, followed by the class of the person. The name contains up to 30 lowercase characters. The class is a string consisting of a nonempty sequence of up to 10 of the words `upper`, `middle`, `lower` separated by hyphens (-), followed by a space, followed by the word `class`. No two people will have the same name in a single case.

Output

For each test case, print the list of names from highest to lowest class. If two people have the same or equivalent classes, they should be listed in alphabetical order by name. Output a line of 30 equal signs (=) after each case.

Sample Input

```
1
5
mom: upper-upper-lower-middle class
dad: middle-middle-middle-lower-middle class
queenelizabeth: upper-upper-upper class
chair: lower-lower class
unclebob: middle-middle-lower-middle class
```

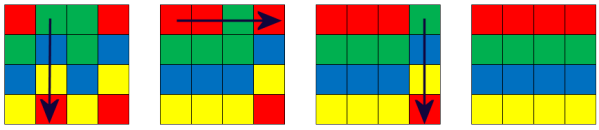
Sample Output

```
queenelizabeth
mom
dad
unclebob
chair
==============================
```

Problem G
Rubik's Revenge in ... 2D!? 3D?
Problem ID: rubiksrevenge
Time Limit: 33 seconds

You are given a puzzle that can be represented as a 4×4 grid of colored cells. The solved puzzle contains 4 monochromatic rows, in this order: red, green, blue, yellow. Although we will analyze this puzzle using its 2D representation, it is actually a 3D puzzle! Imagine that the grid is stretched over a torus (in other words, top edge is connected to the bottom one and left edge is connected to the right one). If you are not familiar with the word "torus" or what it is supposed to represent, just replace it with the word(s) "donut (with the hole in the middle)".

For each move you are allowed to either move one row left or right, or one column up or down. The fact that the outer edges are connected means that if a cell is "pushed out" of the grid, it will reappear on the other side of the grid. If you had a torus or a donut handy (or a cup! HAHAha...ha... <sniff>), this would be much clearer.



Given a description of a state of this puzzle, what is the minimum number of moves you need to solve it? Note that all possible puzzle configurations are solvable in less than 13 moves.

Input

Input file contains exactly 4 lines, containing 4 characters each, each character being either "R", "G", "B" or "Y'. The input will describe a valid state of the puzzle.

Output

Output the minimum number of moves needed to solve the given puzzle.

Sample Input	Sample Output
RGGR GBGB BYBY YRYR	3

Sample Input	Sample Output
RRRR GBGG GYBB BYYY	4

This page is intentionally left blank.

Problem H
The Magical 3
Problem ID: magical3
Time Limit: 3 seconds

There's no doubt about it, three is a magical number. Two's company, but three's a crowd, no one ever talks about 2 blind mice, and there are three members in an ACM ICPC team.

Even more magically, almost all integers can be represented as a number that ends in 3 in some numeric base, sometimes in more than one way. Consider the number 11, which is represented as 13 in base 8 and 23 in base 4. For this problem, you will find the smallest base for a given number so that the number's representation in that base ends in 3.

Input

Each line of the input contains one nonnegative integer n. The value $n = 0$ represents the end of the input and should not be processed. All input integers are less than 2^{31}. There are no more than $1\,000$ nonzero values of n.

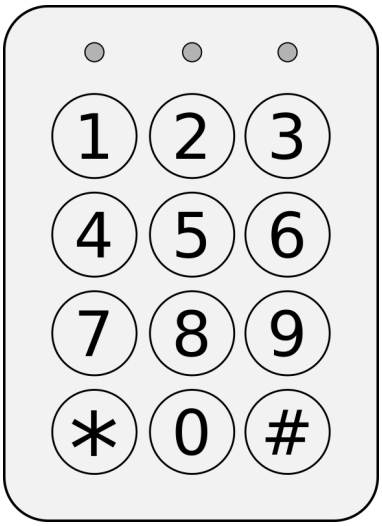
Output

For each nonzero value of n in the input, print on a single line the smallest base for which the number has a representation that ends in 3. If there is no such base, print instead "`No such base`".

Sample Input	Sample Output
11 123 104 2 3 0	4 4 101 No such base 4

This page is intentionally left blank.

Problem I
Matrix Keypad
Problem ID: keypad
Time Limit: 1 second



A matrix keypad consists of an $r \times c$ grid of buttons. Additionally, there is one wire for each row and one wire for each column. These wires are exposed through pins so the keypad can be connected to a larger circuit.

When a button at row i and column j is pressed, the wire for row i and the wire for column j will carry an electrical current. If just a single button is pressed, it can be identified by sequentially checking if a current can be detected at each row wire and at each column wire.

Unfortunately, when multiple buttons are pressed at the same time, it may not be possible to uniquely identify which buttons are pressed. The only information you can have is this: for each wire, whether there is at least one button along that wire being pressed.

The software you are using to detect which buttons are pressed was poorly implemented. After probing the keypad, it stores the information in an $r \times c$ grid of `0/1` values. The value stored in row i and column j of this grid is `1` if there is at least one button in row i and at least one (possibly different) button in column j that is pressed. Otherwise, the value that is stored at this position is `0`.

Your job is to interpret as much information from such a grid as possible. Determine which buttons are definitely pressed and which buttons are definitely not pressed.

Input

The first line of input contains a single positive integer $T \leq 200$ indicating the number of test cases. The first line of each test case contains two integers r and c where $1 \leq r \leq 10$ and $1 \leq c \leq 10$. This indicates that the keypad is an $r \times c$ grid of buttons.

The remaining r lines of a test case describe the grid. The ith row contains a string of consecutive `0` and `1` characters. These will not be separated by spaces.

Output

For each test case, output the following. If there is no combination of button presses on the keypad that would produce this `0/1` grid then simply output a line containing the word `impossible`

Otherwise, you should output r lines, each containing a string of length c. This should describe a grid where the character at row i and column j is:

- `N` if no button combination that produces the input grid has the jth button on row i being pressed.

- `P` if all button combinations that produce the input grid have the jth button on row i being pressed.
- `I` if some, but not all, button combinations that produce the input grid have the jth button on row i being pressed.

Finally, the last line of each test case should be followed by the string `----------` (10 dashes).

Sample Input	**Sample Output**
3 4 3 110 000 110 000 2 3 101 000 2 2 10 01	IIN NNN IIN NNN ---------- PNP NNN ---------- impossible ----------

Problem J
I've Been Everywhere, Man
Problem ID: everywhere
Time Limit: 1 second

Alice travels a lot for her work. Each time she travels, she visits a single city before returning home.

Someone recently asked her "how many different cities have you visited for work?" Thankfully Alice has kept a log of her trips. Help Alice figure out the number of cities she has visited at least once.

Input

The first line of input contains a single positive integer $T \leq 50$ indicating the number of test cases. The first line of each test case also contains a single positive integer n indicating the number of work trips Alice has taken so far. The following n lines describe these trips. The ith such line simply contains the name of the city Alice visited on her ith trip.

Alice's work only sends her to cities with *simple* names: city names only contain lowercase letters, have at least one letter, and do not contain spaces.

The number of trips is at most 100 and no city name contains more than 20 characters.

Output

For each test case, simply output a single line containing a single integer that is the number of distinct cities that Alice has visited on her work trips.

Sample Input	Sample Output
2 7 saskatoon toronto winnipeg toronto vancouver saskatoon toronto 3 edmonton edmonton edmonton	4 1

Problem K
Bundles of Joy
Problem ID: bundles
Time Limit: 3 seconds

Bob's Bakery is celebrating its grand opening! To commemorate this exciting occasion, they are offering a "Bundles of Joy" sale to encourage people to sample their full range of delectable desserts.

For example, you can buy the "Chocolate Cakes" bundle which includes chocolate layer cake and black forest cake for $20. Or you can buy the "Fruity Cakes" bundle which includes lemon pound cake and key lime cake, also for $20. They offer an even bigger bundle that includes a slice of each of these cakes for an even lower price of $38.

You want to try out each dessert they offer. So, you need to buy some bundles to ensure you get at least one of each dessert. Of course, your goal is to do this while minimizing the amount of money you spend on bundles.

Finally, you make a few observations about the bundles they offer:

- For any two bundles A and B, either every dessert in A is also in B, every dessert in B is also in A, or there is no dessert in both A and B.
- The only way to buy an item individually is if it is in a bundle of size 1. Not all items are in such a bundle.
- The pricing is not very well thought out. It may be cheaper to acquire items in a bundle B by buying some combination of other bundles rather than B itself.

Input

The first line contains a single integer $T \leq 50$ indicating the number of test cases. The first line of each test case contains two integers n and m where n is the number of different types of desserts offered by Bob's Bakery and m is the number of different bundles. Here, $1 \leq n \leq 100$ and $1 \leq m \leq 150$.

Then m lines follow, each describing a bundle. The ith such line begins with two positive integers p_i and s_i. Here, $0 < p_i \leq 10^6$ is the price of bundle i and $1 \leq s_i \leq n$ is the number of items in bundle i. The rest of this line consists of s_i distinct integers ranging from 1 to n, indicating what desserts are included in this bundle.

Each of the n items will appear in at least one bundle.

Output

The output for each test case is a single line containing the minimum cost of purchasing bundles to ensure you get at least one of each item. This value is guaranteed to fit in a 32-bit signed integer.

Sample Input	Sample Output
4 4 3 20 2 1 2 20 2 3 4 38 4 1 2 3 4 2 3 5 1 1 10 2 1 2 4 1 2 2 2 1 1 1 5 2 2 1 1 2 2 1 1 1 1 1	38 9 5 1