NWERC 2005

The 2005 ACM Northwestern European Programming Contest
KTH - Royal Institute of Technology, Stockholm, Sweden

The Problem Set

Almost blank page

Problem A: Unequalled Consumption

The Association of Candy Makers is preparing to launch a new product. Its idea is old with a novel twist: it simply sells boxes of candies. But since people are what they consume and everyone wants to be unique these days, the ACM wants *every* candy box to be unique, in the sense that no two boxes should contain the same composition of candy types.

The ACM is only able to make a small number n of different types of candy, but while limited in imagination, it is virtually limitless in resources, so it is able to produce as many as it wants of each type of candy. Furthermore, the candy types have different weights (though some may weigh the same), and in order to simplify pricing matters, the ACM wants all candy boxes to have the same total weight.

With these restrictions, the ACM will only be able to make a limited number of boxes. For instance, if there are three types of candy, weighing $5, 5$ and 10 grams respectively, 4 different boxes can be made with total weight 10 grams (using either two of type 1, or two of type 2, or one of type 3, or one each of types 1 and 2). The ACM would like to be able to make at least one box for everyone in the cosmos. So, given queries in the form of the number of people P in the cosmos, your job is to find the smallest possible total weight w such that P different boxes containing exactly w grams of candies can be made.

Input

The input consists of several data sets (at most 20). Each data set consists of four lines. The first line contains an integer $1 \leq n \leq 5$, the number of candy types. The next line contains n integers $w_1, \ldots, w_n$, where $1 \leq w_i \leq 10$ is the weight (in grams) of the i:th candy type. The third line contains an integer $1 \leq q \leq 10$, the number of queries. The last line of a data set contains q integers $P_1, \ldots, P_q$, where $1 \leq P_j \leq 10^{15}$ is the j:th query. Input is terminated by an incomplete data set where $n = 0$, which should not be processed.

Output

For the i:th data set, write a line "`Set i`", followed by q lines giving, for each query P_j, the minimal possible positive weight W_j (in grams) of a candy box. If there is no weight W_j such that at least P_j candy boxes can be made, print "`no candy for you`" for that query. You may assume that W_j, if it exists, will be at most $100 \cdot P_j$.

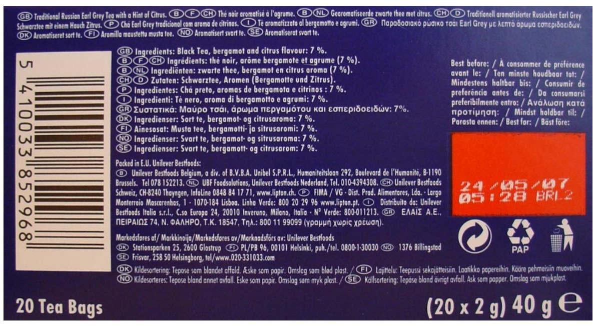
Sample input

```
3
5 5 10
1
4
4
3 1 4 2
2
142 700
1
10
1
100
0
```

Sample output

```
Set 1
10
Set 2
23
42
Set 3
no candy for you
```

Problem B: Declaration of Content



Most food you can buy at your local grocery store has a declaration of content. The declaration of content lists the ingredients of the product. It does not necessarily tell you the exact amount of every ingredient, only the ordering of the ingredients, from most common to least common. For some ingredients, an exact percentage might be given, either required by law or because the producer wants you to know how much of the fine expensive ingredients they have used.

Given a set of different products and their respective declarations of content you should determine which contain the most or the least of some given ingredients. For simplicity, we assume in this problem that the percentage of each ingredient always is an integer.

Input

The input consists of several test cases. Each test case consists of two parts.

The first part of a test case begins with an integer P, $1 \leq P \leq 10$, the number of different products in this test case, on a line of its own. Then follows the description of the P products. Each product description consists of a line giving the name of the product, followed by a line containing an integer n, $1 \leq n \leq 100$, giving the number of ingredients in this product. Then follow n lines, the i:th of which contains the name of the i:th most common ingredient of the product. In case of ties, the ingredients will be listed in arbitrary order. Optionally, every ingredient name can be followed by space, an integer p, $0 \leq p \leq 100$ and a percentage sign. If this is present, it specifies the exact amount of this ingredient in the product. Otherwise, because all percentages in this problem are integers, the ingredient makes up at least one percent of the total product.

The second part of a test case begins with an integer Q, $1 \leq Q \leq 100$, the number of queries. Then follow Q lines, each containing a query. A query is of the form "least X", or "most X", where X is the name of an ingredient. In the "most X" case, the ingredient X is guaranteed to be present in at least one of the products.

A name of a product or an ingredient is a string of alphabetic characters (A-Z and a-z), digits (0-9) and underscore. Case is significant. No name will be longer than 30 characters. You may assume that each declaration of content is valid.

The last test case to be processed is followed by a line consisting of the integer 0.

Output

The output consists of one line for every query in the input data. For each query, output the name of the product containing the most or the least of ingredient X, as indicated by the query. If there are several possible such products, output all of them, in the same order as the products were presented in the test case input data. The product names should be separated by a single space.

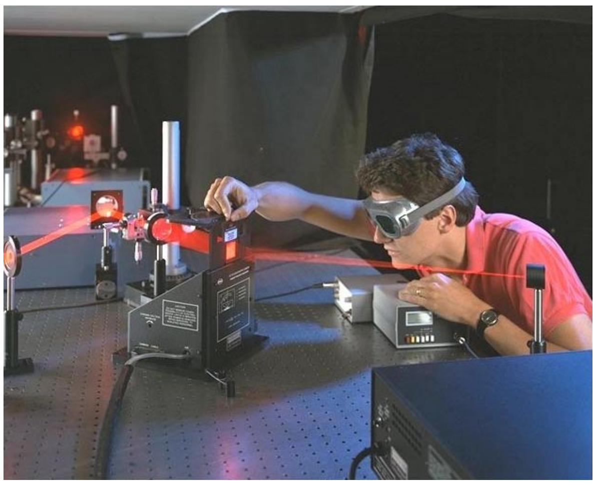
Sample input

```
3
Product_1
3
A
B
C
Product_2
3
C
B
A
Product_3
2
B
C 35%
4
most A
most B
most C
least D
0
```

Sample output

```
Product_1
Product_3
Product_2 Product_3
Product_1 Product_2 Product_3
```

Problem C: Laserbox



A laserbox is a game involving some optical equipment. The game board is a square $n \times n$ grid. On each grid point, a gadget called a right-turner can be placed and several such gadgets are included. Finally, there is a ruby laser, and if the laser is mounted at the bottom end of a column, the beam will be directed northwards through that column. Analogously, the laser beam may be directed southwards from the top of a column, eastwards from the start of a row or westwards from the end of the row.

The game starts with some right-turners being spread out on some grid points and the laser (switched off) being mounted somewhere along the border of the rectangle. The player then tries to deduce where the beam will emerge when the laser is switched on. The effect of a right-turner is to deflect the beam ninety degrees to the right, regardless of from which of the four directions it enters.

Your program must do exactly what the player is supposed to do.

Input

On the first line of the input is a single positive integer, telling the number of test cases to follow. The first line of each test case consists of two integers n r, where $1 \leq n \leq 50$ is the size of the board and $1 \leq r \leq 50$ the number of right-turners. The following r lines contain the coordinates x y of the right-turners. No two right-turners will have the same coordinates.

Finally, a line with two integers indicating the laser position follows. The bottom of column six is denoted by 6 0 and the start of row seven by 0 7. If the zeroes are replaced by $n + 1$, the laser is placed at the top of column six and the end of row seven, respectively.

Output

For each test case, output one line containing the coordinates X Y of the beam as it leaves the board. The same rules as for the laser apply, so X may equal 0 or $n + 1$ or else Y equal 0 or $n + 1$. If the beam gets caught and does not leave the board, the output should be 0 0.

Sample input

```
2
2 3
1 1
1 2
2 2
3 1
3 6
1 1
1 3
2 2
2 3
3 1
3 2
2 0
```

Sample output

```
2 0
0 2
```

Problem D: Bowlstack

Baking bread is my favourite spare-time pursuit. I have a number of stainless steel mixing bowls with straight sides, a circular bottom and a wider circular top opening. Geometrically, my bowls are truncated circular cones and for this problem, the thickness of the metal may be disregarded.

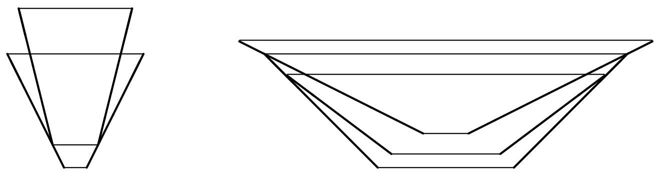
I store these bowls stacked in the natural way, that is with a common vertical axis, and I stack them in an order that minimises the total height of the stack. Finding this minimum is the purpose of your program.

Input

On the first line of the input is a positive integer, telling the number of test cases to follow. Each case starts with one line containing an integer n, the number of bowls ($2 \leq n \leq 9$). The following n lines each contain three positive integers h, r, R, specifying the height, the bottom radius and the top radius of the bowl, and $r < R$ holds true. You may also assume that $h, r, R < 1000$.

Output

For each test case, output one line containing the minimal stack height, truncated to an integer (note: truncated, not rounded).



Sample input	**Sample output**

`2`
`2`
`60 20 30`
`40 10 50`
`3`
`50 30 80`
`35 25 70`
`40 10 90`
	`70`
	`55`

Almost blank page

Problem E: Pesky Heroes

“Pesky Heroes!” — the evil mage stormed into his throneroom — “They don’t have *any* respect for us evil masterminds anymore. They’ve gone and invaded my cave again, and while the traps were down for maintenance! And they’ve set up their camp in the cave! The nerve!”

“But, milord, then surely they will be doomed when the traps have been reactivated,” were the last words of a servant, uttered milliseconds before he. . . well, you get the picture. “Not necessarily, I may have to teleport my trained orcs to seal their fate. You there, figure out where I should send them. Now!”

Inspired by the fate of the last servant to speak, you hurry off with a map of the cave. The map is a complete map of the cave, with the positions of all traps marked, and you see that the cave has a single entrance. On the map, there are n strategic *key points* which have been marked with the numbers 1 to n. All passages in the cave connect two key points. All traps have been placed at key points, and all dead ends are key points. Your mathematically inclined mind quickly discovers that for every key point there are at most three passages connecting to it.

Your master can open a magic gateway from the throneroom to any passage (but not to key points), and the trained orcs will be sent in. Despite their long training, the orcs are still not too bright. When they’ve gone through the gateway, they will begin walking away from the light coming from the entrance (there will always be a unique path from each point in the cave to the entrance). In each intersection, they will normally always choose the left path, and when they come back to the gateway, they will come back to the throneroom, ready to be teleported to a new location, if necessary.

These orcs have been trained, so they’re smarter than the average bear, erm, orc. You can give them a number t when they walk in, and the t:th time they are at an intersection, they will turn right instead of left. Despite their eight years of training, they can still only be trusted to remember one number until they return.

“Oh, and one last thing. I saw the heroes in my crystal ball, they’ve set up camp in a dead-end.”, the mage shouts after you.

To maximize your chances of survival, you want your master to have to work as little as possible (i.e. to open as few gateways as possible), but you must have the orcs search every dead-end in the cave which can reach the entrance without passing any traps. The orcs are very dear to your master, so you better make sure they don’t walk into any traps!

Input

The input will consist of several data sets. Each set will start with a line consisting of two numbers, n, m where $2 \leq n \leq 50000$ is the number of key points on the map and $0 \leq m \leq 500$ is the number of traps.

The next n lines will consist of 2 to 4 space-separated integers. Line i contains an integer n_i, $1 \leq n_i \leq 3$, the number of passages connecting to key point i, followed by a list of the n_i key points that the passages lead to, in clockwise order. The next m lines consist of single integers, the key points at which there are traps. Key points are labelled $1, \ldots, n$ and key point 1 is the (implicit) entrance to the outside world. Key point 1 is guaranteed to always have exactly one passage.

The last case will be followed by a line with $m = n = 0$, which should not be processed.

Output

For each case, output the minimum number of gateways required, on a line by itself.

Sample input

```
5 1
1 2
3 1 4 3
1 2
2 2 5
1 4
3
4 1
1 2
2 1 3
2 2 4
1 3
4
0 0
```

Sample output

```
1
0
```

Problem F: Reduced ID Numbers

T. Chur teaches various groups of students at university U. Every U-student has a unique Student Identification Number (SIN). A SIN s is an integer in the range $0 \leq s \leq MaxSIN$ with $MaxSIN = 10^6 - 1$. T. Chur finds this range of SINs too large for identification within her groups. For each group, she wants to find the smallest positive integer m, such that within the group all SINs reduced modulo m are unique.

Input

On the first line of the input is a single positive integer N, telling the number of test cases (groups) to follow. Each case starts with one line containing the integer G ($1 \leq G \leq 300$): the number of students in the group. The following G lines each contain one SIN. The SINs within a group are distinct, though not necessarily sorted.

Output

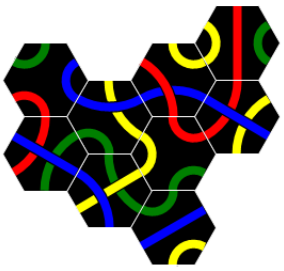
For each test case, output one line containing the smallest modulus m, such that all SINs reduced modulo m are distinct.

Sample input	**Sample output**

`2`
`1`
`124866`
`3`
`124866`
`111111`
`987651`
	`1`
	`8`

Almost blank page

Problem G: Tantrix

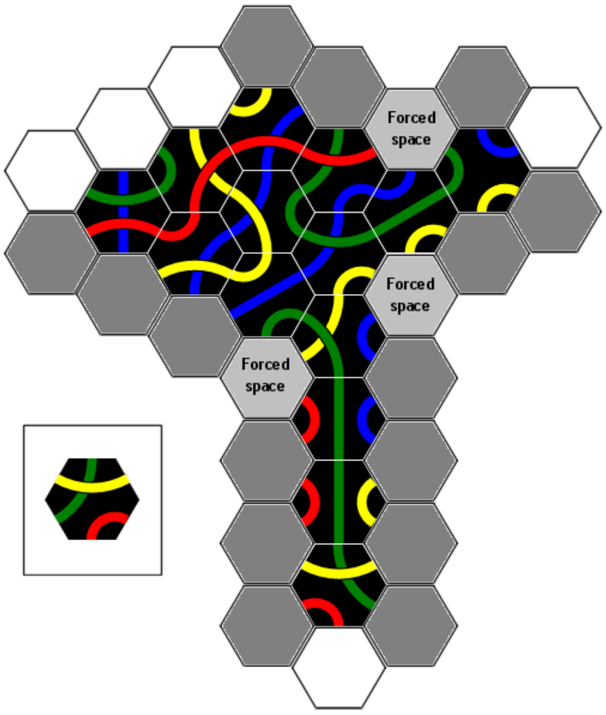


Tantrix is a two player game played with 56 hexagonal tiles. Each tile contains three *links* in different colours. Both players have five tiles in hand and take turns in placing them on the playing field. The figure to the right shows how the game could have progressed after nine played tiles.

There are four different link colours: red, green, yellow and blue. No two tiles are identical, and no tile is rotation symmetric. A tile will be described in the input as a six letter string, specifying the link colours in clockwise direction. The uppercase letters 'R', 'G', 'Y' and 'B' will be used for red, green, yellow and blue, respectively.

In this problem, a move is defined as placing one of the tiles in hand somewhere on the playing field, subject to these rules:

1. A tile must always be placed next to tiles already played.
2. The links in all touching tiles must match colour.
3. An empty space which is surrounded by three tiles is called a *forced space*. If the player can place one of his tiles in a forced space, he must do so. If there are several forced spaces, and several ways to place a tile in a forced space, he may select any of those.
4. It's not allowed to place a tile so that a forced space is created containing three links of the same colour (since no tile could ever be placed there).
5. The two sides along a forced space are called *controlled sides*. It's not allowed to place a tile along a controlled side.



If there are one or more forced spaces and the player can't place any of his tiles in hand in those spaces, he will have to play any other legal move. Note that a player may not be allowed to place a tile in a forced space due to rule 4.

The figure on the right illustrates these rules. There are three forced spaces. The interposed tile may not be placed in the lower left forced space, as that would create a new forced space with three red links. The dark gray spaces lie on controlled sides created by the forced spaces; no tiles may be placed there. If the player to move can't place a tile in any of the three forced spaces, he must place a tile in any of the white spaces.

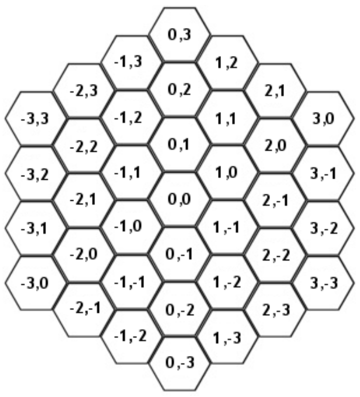
Your task is to count the number of legal moves the player to move has, given the position and orientation of already played tiles and the tiles in hand for the player to move. If a tile can be placed at several locations, or in several orientations, each such combination is counted as a distinct move.

Input

The first line in the input will contain the number of cases (at most 50).

Each case begins with a single line containing an integer n ($1 \leq n \leq 20$), the number of tiles that have already been played. Then follow n lines containing the coordinates and description of these tiles. The first character in the tile description belongs to the link facing up; the remaining colours follow as per usual in clockwise direction. Then follows a line with the description of the five tiles in hand, the tile descriptions being separated with a single space.

The mapping between the spaces and the coordinates is shown in the figure below (note that the playing field is infinite and not restricted to these coordinates). All tiles in the input will be valid and distinct. The layout will represent a position that could have arisen from a legal game. One of the played tiles will have coordinates $0, 0$.



Output

For each test case, output a single line containing an integer: the number of legal moves.

Sample input

```
2
6
0 0 BRYRBY
1 0 GRGBRB
-1 1 GGYBYB
0 1 YYBBGG
-2 2 YYBGBG
-3 3 BYGYGB
BBRRGG GBYBYG RBRBGG GYBGBY GRBBRG
4
0 0 BYYGBG
-1 1 GRGBBR
1 0 YRBRYB
2 0 YGGRRY
RBBRYY GBGYBY YBBRYR YBYBRR RBBRGG
```

Sample output

```
46
2
```

Problem H: Guardian of Decency

Frank N. Stein is a very conservative high-school teacher. He wants to take some of his students on an excursion, but he is afraid that some of them might become couples. While you can never exclude this possibility, he has made some rules that he thinks indicates a low probability two persons will become a couple:

- Their height differs by more than 40 cm.
- They are of the same sex.
- Their preferred music style is different.
- Their favourite sport is the same (they are likely to be fans of different teams and that would result in fighting).

So, for *any two* persons that he brings on the excursion, they must satisfy *at least one* of the requirements above. Help him find the maximum number of persons he can take, given their vital information.

Input

The first line of the input consists of an integer $T \leq 100$ giving the number of test cases. The first line of each test case consists of an integer $N \leq 500$ giving the number of pupils. Next there will be one line for each pupil consisting of four space-separated data items:

- an integer h giving the height in cm;
- a character `'F'` for female or `'M'` for male;
- a string describing the preferred music style;
- a string with the name of the favourite sport.

No string in the input will contain more than 100 characters, nor will any string contain any whitespace.

Output

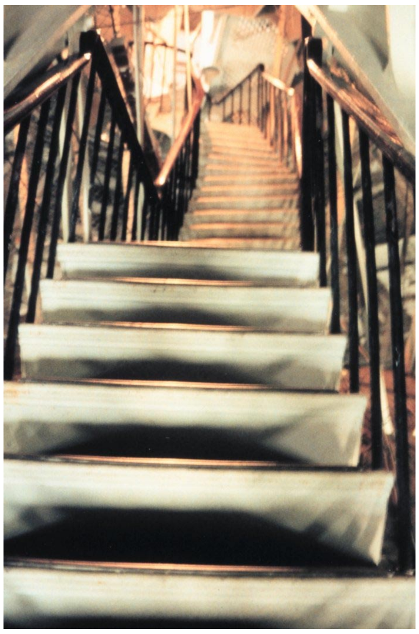
For each test case in the input there should be one line with an integer giving the maximum number of eligible pupils.

Sample input	Sample output

`2`
`4`
`35 M classicism programming`
`0 M baroque skiing`
`43 M baroque chess`
`30 F baroque soccer`
`8`
`27 M romance programming`
`194 F baroque programming`
`67 M baroque ping-pong`
`51 M classicism programming`
`80 M classicism Paintball`
`35 M baroque ping-pong`
`39 F romance ping-pong`
`110 M romance Paintball`
	`3`
	`7`

Problem I: Up the Stairs

John is moving to the penthouse of a tall sky-scraper. He packed all his stuff in boxes and drove them to the entrance of the building on the ground floor. Unfortunately the elevator is out of order, so the boxes have to be moved up the stairs.



Luckily John has a lot of friends that want to help carrying his boxes up. They all walk the stairway at the same speed of 1 floor per minute, regardless of whether they carry a box or not. The stairway however is so narrow that two persons can't pass each other on it. Therefore they deciced to do the following: someone with a box in his hands is always moving up and someone empty-handed is always moving down. When two persons meet each other somewhere on the stairway, the lower one (with a box) hands it over to the higher one (without a box). (And then the lower one walks down again and the higher one walks up.) The box exchange is instantaneous. When someone is back on the ground floor, he picks up a box and starts walking up. When someone is at the penthouse, he drops the box and walks down again.

After a while the persons are scattered across the stairway, some of them with boxes in their hands and some without. There are still a number of boxes on the ground floor and John is wondering how much more time it will take before all the boxes are up. Help him to find out!

Input

One line with a positive number: the number of test cases. Then for each test case:

- One line with three numbers N, F, B with $1 \leq N, F \leq 1,000$ and $1 \leq B \leq 1,000,000$: the number of persons, the number of floors (0=ground floor, F=penthouse) and the number of boxes that are still on the ground floor.

- N lines with two numbers f_i and b_i with $0 \leq f_i \leq F$ and $b_i = 0$ or $b_i = 1$: the floors where the persons are initially and whether or not they have a box in their hands (1=box, 0=no box).

Output

One line with the amount of time (in minutes) it will take to get all the remaining boxes to the penthouse.

Sample input

```
2
3 10 5
0 0
0 0
0 0
2 5 1
2 1
3 0
```

Sample output

```
30
8
```