Problem A – Interstellar

We are in future, 2050, where humans have been conquered the space and people are spread out between different planets. For having a nice-looking space, all planets have been moved in a way that if you look at the new space from sun, you'll see a cube of planets each laid on a positive integer coordination in the cube. The ordered planets are bounded in a X*Y*Z cube. Note that in each coordination there is exactly one planet.
Visiting your friends are now even harder and more expensive than any time and you need to have inter-planet flights with jets. Since having high-speed jets really matters in space trips, commercial jets have high-speed (half of speed of light!) but with one limitation that they cannot change their direction and can move only in one direct line between their source and destination. Traditionally, if your jet passes another planet during the trip, you have to pay 1 planeto (new interstellar currency) “space toll road”. For example if you want to trip from (1,1,1) to (5,9,5) you have to pay 3 planetos since you'll visit planets, (2,3,2), (3,5,3), (4,7,4).

Given X, Y and Z, the size of cube of planets, we are interested to know the summation of planetos that will be paid, if we travel between all pairs of planets.

Input

In first line of input there is T, number of test cases.
Each test case contains three positive integers, X, Y, Z. Its guarantee that the number of planets is less than 10^6.

Output

For each output print the summation of planetos will be paid by visiting every pair of planets modulo 1000000007.

Sample Input

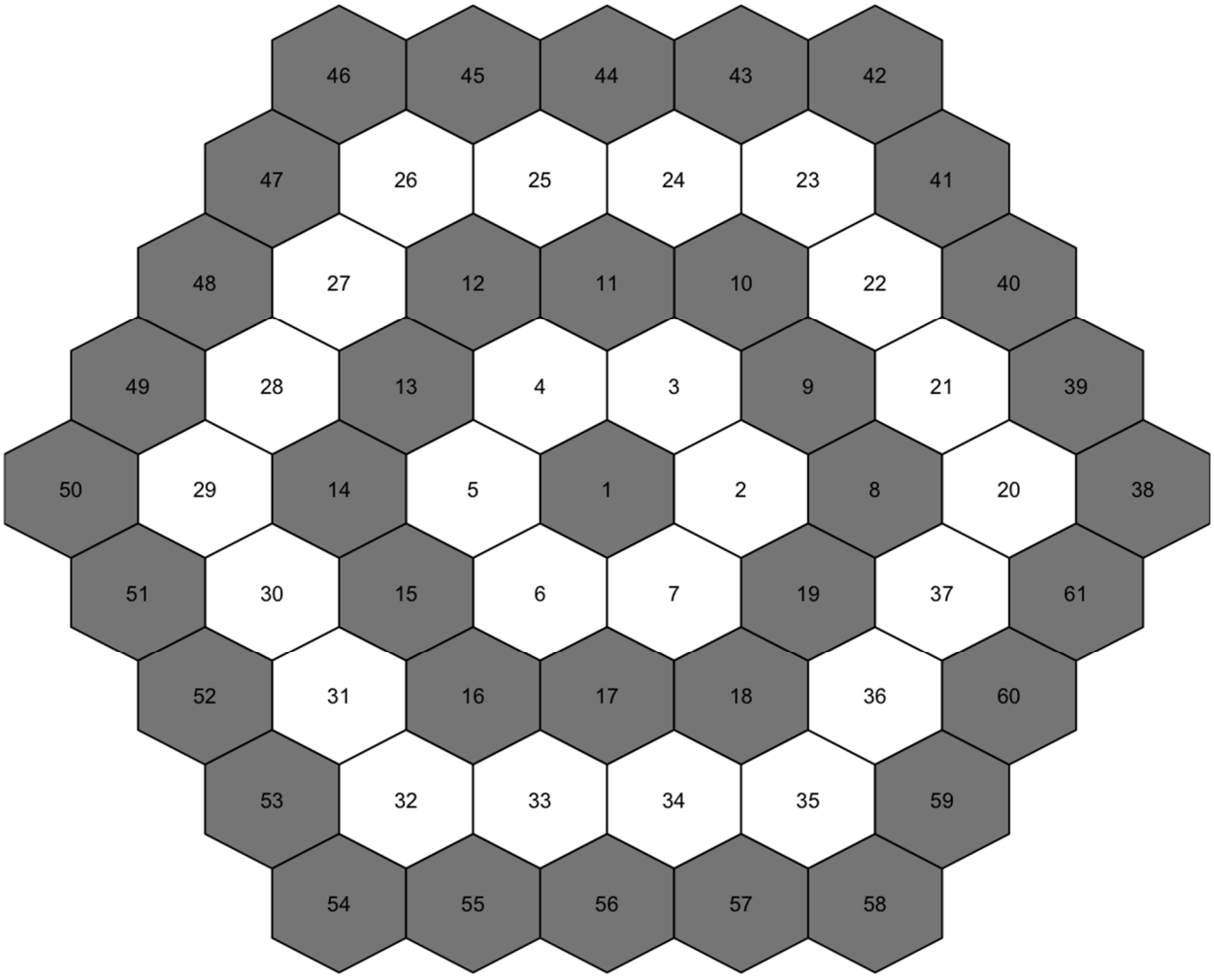
```
3
1 2 1
3 1 1
12 5 7
```

Sample Output

```
0
1
17438
```

Problem B – Hexagonal connectivity

Having a hex world which some tile has a color of Red, Blue, Green or Gray and rest of them have no color (or white). We want to know what is the minimum number of colors that we have to walk to get from tile number A to tile number B. Note that we cannot walk over tiles which have no color (or white tiles).



The indexing system is like above.

Input

Input consists of several test cases, each test case is started with two integers, **N** and **M**. **N** representing the number of tiles with color and **M** the number of paired tiles which we want to know the minimum colors for getting from one to another.

Then there are **N** line, each having a tile number and a color (**red, blue, green, gray**), separated by "-". And after that there are **M** lines each having 2 tile indexes (which both were noticed in the **N** lines) **N** would be less than 100000.

Output

For each 2 tiles, input 0 if there is no path and if there is a path, output the minimum number of colors needed to get through.

Sample Input

```
5 3
1-gray
2-gray
8-gray
37-green
60-gray
1 60
1 37
7 8
```

Sample Output

```
2

2

0
```

Problem C – First name

For choosing our first names, many of our ancestors didn't consider its length. For a considerable number of people their first name doesn't fit on their passports or bank cards. Sometimes they have to change them or make them shorter.
Recently one of the governments decided to shorten all first names of its citizens. To do so, government mandates that there should not be any repeated characters in new first names. After announcing this rule, they realized that people started picking the easiest first names such as "ya", "sh", "f", "ir" etc. Thus, they added one more rule to avoid having too many similar first names. The new rule says new first names must contain all different type of characters as their original one and also the order of characters cannot be changed. For example, "babak" could be changed to "bak" but not "abak" or "kabab" or "bz".
After announcing these rules, people started thinking to how to change their first name in order to have the smallest lexicographical possible one.

Given first names, for each one find the smallest lexicographical new first name based on aforementioned rules.

Input

In first line of input there is T, number of test cases.
For each test case, there is one line of input which contains string S which constists of small letters of English and its size is less than 20.

Output

Foe each test case, print the smallest possible first name on one line.

Sample Input

```
3
babak
zelatan
khajeshamsoddin
```

Sample Output

```
abk
zelatn
kajehmsodin
```

Problem D - Last name

For choosing our last names, many of our ancestors didn't consider its length. For a considerable number of people their last names don't fit on their passports or bank cards. Sometimes they have to change them or make them shorter.
Recently one of the governments decided to shorten all last names of its citizens. To do so, government mandates that there should not be any repeated characters in new last names. After announcing this rule, they realized that people started picking the easiest family names such as "a", "ap", "bz", "bd" etc. Thus, they added one more rule to avoid having too many similar family names. The new rule says new last names must contain all different type of characters as their original one and also the order of characters can not be changed. For example, "zanjani" could be changed to "zanji" or "zajni" but not "zanjai" or "jazani" or "bz".
After announcing these rules, people started thinking to how to change their last name in order to have the smallest lexicographical possible last name.

Given last names, for each one find the smallest lexicographical new last name based on aforementioned rules.

Input

In first line of input there is T, number of test cases.
For each test case, there is one line of input which contains string S which consists of small letters of English and its size is less than or equal to 200000.

Output

Foe each test case, print the smallest possible last name on one line.

Sample Input

```
3
zanjani
ibrahimovic
hafezshirazi
```

Sample Output

```
zajni
brahimovc
afeshirz
```

Problem E – Edges of that thing

Maybe you have heard about that problem where you have a set of nails on a board and you put a rubber band around them and release it, the rubber shrinks and get stuck on some nails. Then you want to find out something about the shape of that rubber band (like the set of nails that it is touching, size of band, etc).

But that's easy, so this problem ain't that nails on a board. Let's assume that instead of having a set of nails on a board, we have a set of fixed points in the air (I couldn't figure out how it works in real world, but feel free to imagine for yourself), and instead of a rubber band, we have a rubber ball, that is surrounding all these points. Now what happens if we release the ball? What would be the shape of it? In general it would form a polyhedron. Picturing that in mind, I was wondering for a specific set of points, how many edges that thing has! Therefore, your task is to answer my wonder.

Input

The input starts with an integer T, denoting the number of test cases.

Each test case starts with an integer n (8 <= n <= 100), which represents the number of points. Then comes n lines, each having three space-separated integers x, y and z, which are the coordinates of each point in the air.

Output

For each test case, output one integer, the number of edges on that thing!

Sample Input

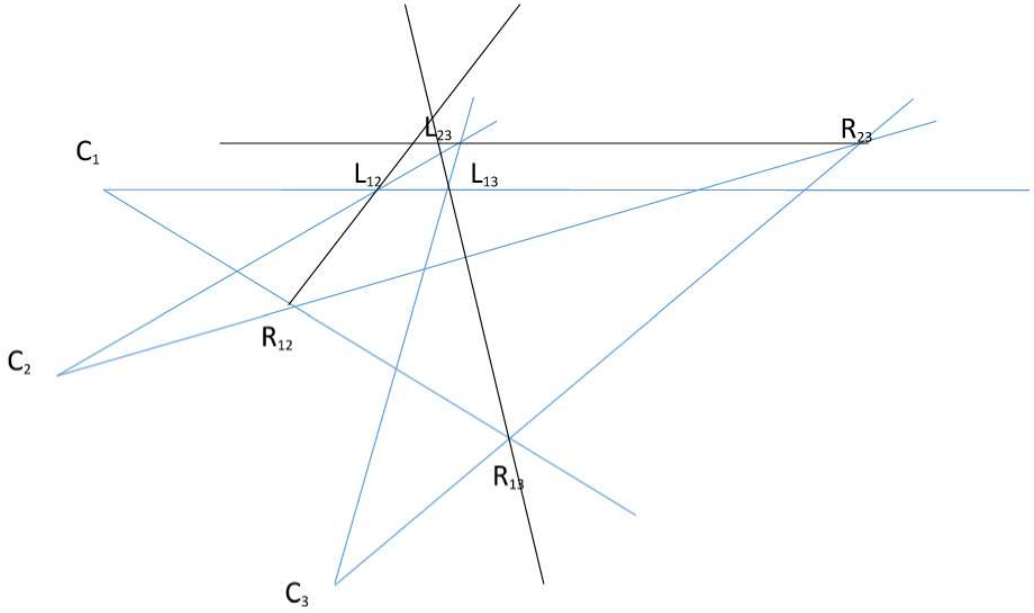
```
2
10
0 0 0
2 0 0
2 2 0
0 2 0
0 0 2
2 0 2
2 2 2
0 2 2
1 1 1
1 2 1
8
0 0 0
4 0 0
4 4 0
0 4 0
2 2 0
2 2 1
2 2 2
2 2 4
```

```
Sample Output
12
8
```

Note: the first test case is a cube and the second test case is a pyramid with a square base (four edges in base and four edges connecting the top to the base).

Problem F – Special Camera Linearity

Having 3 cameras, we call these cameras Linear, if they fulfill the below condition.
Imagine C1, C2 and C3 are the 3 cameras, each one having their angle of view. We call the Intersection of the Left view line of C1 and the same line of C2, L12, and Right view line of C1 and the same line of C2, R12 and like that we create 6 intersections, L12, L13, L23, R12, R13 and R23 . As it can be seen, L12R12 is the only line of view that is common in the view of C1 and C2. Lines of views are L12R12, L13R13, L23R23. If these three lines have the same point of intersection, we call these 3 cameras special Linear.



Knowing this definition, we have N cameras, all with a degree view of 30 degrees and random rotation pattern which ensures all possible cross overs between these cameras, your job is to calculate the number of three triplet cameras that are linear.

Input

Input consists of several test cases. Each test case starts an Integer N, the number of cameras on the field (which is less than 10000). This integer is followed by N lines, each having 2 integers, the coordinates of cameras (x, y) in which are less than 10000.

Output

For each Input, output the number of triplets that are by the above definition, Linear, in at least one possible cross over.

Sample Input

```
1
4
0 0
1 1
0 1
1 0
6
0 0
0 1
0 2
0 3
1 0
1 1
```

Sample Output

```
0
4
```

Problem G – Final Problem

As you know, we, human beings, invented words to be able to convey our messages to each other in a clear and unambiguous way. But since everything is being changed based on our priorities, we cannot trust to meaning of words anymore. Here we just want to focus on one of them.

In priority-land, these days there is a new definition for word “final”. Based on the new definition, something is final, when it's closest thing to the last thing but not the last one itself!

For example, if we had a list of dates, as six-digit numbers, [940821,940905, 940828,940912,940716], traditionally 940912 would be the final date but priority-wise and based on new definition, 940905 is considered as final date.

Given a stream of dates, your task is to print the final date once in a while.

Input

There are a bunch of queries that end with the word “END”.

Each query could be a date in a six-digit format, which means you need to add this new date to the previous dates, or word “GetFinal”, which means you need to print the final date so far. It's guaranteed that “GetFinal” shows up when there is more than one date.

Output

For each “GetFinal” query, write the final date so far in a line.

Sample Input

```
940821
940905
GetFinal
940828
940912
940716
GetFinal
END
```

Sample Output

```
940821
940905
```

Problem H – Guards at the Museum

There is a museum in town an there are very valuable items in it.
This museum needs to be guarded. The security department wants you to find a placement for the guards which meet all of their conditions. You are given floor plan of museum, which is a square grid with positioning of walls and items. The criteria are:

- A guard stands in a cell and doesn't move. A guard can watch over
- Cells in his row or column (up to a wall or item).
- No two guards should see each other.
- All cells in the museum must be watched by some guards.
- Items need to be closely guarded. Each item has a value between

0 and 4 (inclusive), which shows how many guards, should stand next to that item ("next to here" refer to adjacent horizontally or vertically). An item with 0 value does not need to be guarded and you should avoid putting a guard next to it (we need to keep the number of guards low).

Input

The input starts with an integer T, denoting the number of test cases.
Each test case starts with an integer n (5 <= n <= 25), which denotes the size of museum. Then comes n lines, each having n character, representing the layout of museum.
'.' : Empty space
'#' : Wall
'0-4' : An item (number of guards needed around this cell)

Output

For each test case, output the layout of museum with guards in them,
put an 'x' for a guard (without quotes of course). See the
sample output for how the layout should look like.
It's guaranteed that answers are unique for input layouts.

```
Sample Input:
1
7
...1.0.
#......
..#.#..
#.....#
..#.3..
......#
.3.2...

Sample Output:
..x1.0.
#...x..
x.#.#.x
#...x.#
..#x3x.
.x....#
x3x2x..
```