Ancient Plates

Time Limit:1000MS Memory Limit:30000K

Description

Background

Some decades ago archaeologists discovered plates in an ancient temple that were covered with text. While the symbols used were known to the archaeologists, the encryption of the text as a whole has so far been a mystery. However, just last year another temple was discovered in the same area as the first. The interior of this temple was quite similar to the first, so it is believed to have been built at the same time. The second temple contained plates as well, but the contents of these plates looked quite different. After a year of research, the archaeologists developed the theory that these plates contained the code which was needed to decipher the text on the plates found in the first temple.

Problem

Your job is to write a program which automates the process of deciphering the text on the plates by applying the rules which were proposed by the archaeologists.

There are two types of rules: PERM and SHIFT. Both apply to one of the following targets: lines of the text (L), words (W) or characters (C).

A PERM rule consists of the string PERM, followed by a single blank and a letter indicating the target, and one or more pairs of indices separated by blanks. A pair of indices i, j consists of two positive integers i and j separated by just a comma.

Depending on the target, the PERM rule is supposed to do the following for each pair i, j of indices, working with the indices from the left to the right:

L : exchange lines i and j;

W : exchange words i and j in all lines;

C : exchange characters i and j in all words in all lines.

If at least one of the objects to be exchanged does not exist, skip this exchange.

Example: PERM W 2,7 asks you to exchange the second and seventh word in all lines, but there might be a line with just five words.Then do nothing in that line, but perform the exchange in all other lines with at least seven words.

A SHIFT rule consists of the string SHIFT, a single blank, the letter indicating the target, another single blank and an integer k. If k = 1 [k = -1], you are supposed to do the following, depending on the target:

L : shift all lines up [down] by one line, with the first [last] line becoming the last [first] line;

W: shift all words in all lines to the left [right] by one word, with the first [last] word in each line becoming the last [first] word.

C : shift all characters in all words to the left [right] by one character, with the first [last] character of each word becoming the last [first] character.

If k >= 0 [k <= 0] is an arbitrary integer, perform the above action for k = 1 [k = -1] just |k| times.

Input

The first line of the input contains the number of scenarios, i.e. the number of plates which have to be deciphered.

For each such plate, you are first given the number l (0 <= l <= 1000) of lines of text on the plate to be deciphered in a single line. In the following l lines, this text is given, with no line exceeding 1000 characters. After that, there is a line containing the number r (0 <= r <= 1000) of rules which have to be applied. The subsequent r lines contain these rules, in the format described above.

The words of the text are considered to be sequences of arbitrary characters (not just letters!), separated by single blanks. No line of the given text will start or end with a blank, but it might be empty altogether.

Output

Start the output for each scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print l lines containing the text after applying all the rules. Print an additional blank line after each scenario.

Sample Input

```
2
3
First line.
Second line.
Third line.
1
SHIFT L -1
6
ettypr ante e.aid era a
```

lsemi is an btiingOr itsh at a satncedi of uhglyro nten-twoyi lilonmi rgeurdedan lolwye ns.u setrnwe iarlsp mra of eht lxayGa elsi a allsm oeswh ed-dsceneedap flei trelyut sgiaificnntin tlteli ubel enegr aentpl dne of eht raF tuo in eht cahntedur cwkbtersaa of eht fsabionahleun iknth gtialdi thceswa rsmfo era so aizmglyan iimrivept atth etyh illst
5
SHIFT W 3
PERM C 4,9 2,3
SHIFT C -2
SHIFT L 1
PERM L 3,9 3,4 1,3

Sample Output

Scenario #1:
Third line.
First line.
Second line.

Scenario #2:
Far out in the uncharted backwaters of the unfashionable end of the western spiral arm of the Galaxy lies a small unregarded yellow sun. Orbiting this at a distance of roughly ninety-two million miles is an utterly insignificant little blue green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea.

Box Art

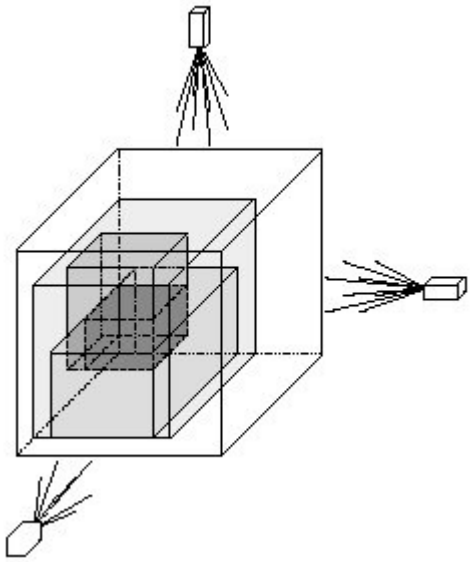
Time Limit:5000MS Memory Limit:262144K

Description

Background

The world famous artist A.A. Blox, well known for his cubic sculptures, has developed a totally new way to create impressive artwork from a rectangular solid of transparent acrylic glass. With the patented laser device of his friend T.D. Resal, he is able to change the colour of parts of the originally colourless box. Due to the prototype stadium of the laser device, he can only change the colour of a rectangular solid whose sides are parallel to the sides of the large box ("axis aligned").

The value of the resulting object is measured by the volume of coloured acrylic glass. Since A.A. Blox is not good at mathematics, he has hired you to help him out and compute the price of his artwork for him.



Problem

For a given three-dimensional axis aligned initial box b and a set S of three-dimensional axis aligned boxes,you have to compute the volume of the union of all parts of the boxes of S that lie within b. Make sure that you count the volume of overlapping parts of the boxes only once!

Input

The first line contains the number of scenarios.

For each scenario you are given a line containing x1 y1 z1 x2 y2 z2, defining the two corners (x1, y1, z1), (x2, y2, z2) of the initial axis aligned box b. All numbers are separated by single blanks.
The following line contains the numberm(m <= 2000) of boxes in S whose colour was changed by the laser device, followed bymlines each containing x1 y1 z1 x2 y2 z2, defining the two corners (x1, y1, z1),(x2, y2, z2) of one of the axis aligned boxes in S. All numbers are separated by single blanks.
All coordinates are in the range from 0 to 1000, and the coordinates in each line satisfy x1 <= x2, y1 <= y2 and z1 <= z2.

Output

Start the output for every scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a line containing the total volume of coloured acrylic glass. Terminate the output for the scenario with a blank line.

Sample Input

```
2
0 0 0 10 10 10
1
2 2 2 4 4 4
0 0 0 10 10 10
2
0 0 0 6 10 10
4 0 0 10 10 10
```

Sample Output

```
Scenario #1:
8

Scenario #2:
1000
```

Brainman

Time Limit:1000MS Memory Limit:30000K

Description

Background

Raymond Babbitt drives his brother Charlie mad. Recently Raymond counted 246 toothpicks spilled all over the floor in an instant just by glancing at them. And he can even count Poker cards. Charlie would love to be able to do cool things like that, too. He wants to beat his brother in a similar task.

Problem

Here's what Charlie thinks of. Imagine you get a sequence of N numbers. The goal is to move the numbers around so that at the end the sequence is ordered. The only operation allowed is to swap two adjacent numbers. Let us try an example:

Start with: 2 8 0 3

swap (2 8) 8 2 0 3

swap (2 0) 8 0 2 3

swap (2 3) 8 0 3 2

swap (8 0) 0 8 3 2

swap (8 3) 0 3 8 2

swap (8 2) 0 3 2 8

swap (3 2) 0 2 3 8

swap (3 8) 0 2 8 3

swap (8 3) 0 2 3 8

So the sequence (2 8 0 3) can be sorted with nine swaps of adjacent numbers. However, it is even possible to sort it with three such swaps:

Start with: 2 8 0 3

swap (8 0) 2 0 8 3

swap (2 0) 0 2 8 3

swap (8 3) 0 2 3 8

The question is: What is the minimum number of swaps of adjacent numbers to sort a given sequence?Since Charlie does not have Raymond's mental capabilities, he

decides to cheat. Here is where you come into play. He asks you to write a computer program for him that answers the question. Rest assured he will pay a very good prize for it.

Input

The first line contains the number of scenarios.

For every scenario, you are given a line containing first the length N (1 <= N <= 1000) of the sequence,followed by the N elements of the sequence (each element is an integer in [-1000000, 1000000]). All numbers in this line are separated by single blanks.

Output

Start the output for every scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing the minimal number of swaps of adjacent numbers that are necessary to sort the given sequence. Terminate the output for the scenario with a blank line.

Sample Input

```
4
4 2 8 0 3
10 0 1 2 3 4 5 6 7 8 9
6 -42 23 6 28 -100 65537
5 0 0 0 0 0
```

Sample Output

```
Scenario #1:
3

Scenario #2:
0

Scenario #3:
5

Scenario #4:
0
```

Friends

Time Limit:1000MS Memory Limit:30000K

Description

You know this problem... several friends meet at the Hacienda (a rather fine restaurant) every Wednesday,have some drinks and fun in general until someone brings up the dreaded question: "So, where are we going tonight?"

At this point basically two things can happen: Either take the Swedish solution or the Italian solution. The Swedish solution means that there is an absolutely embarrassing silence for about five minutes disturbed only once or twice by a muttered "Uh, I don't care...", then they finally agree (silently of course) to get totally drunk. The Italian solution includes (but is not limited to) lots of shouting, total mayhem in the Hacienda, and probably 'family' problems.

But wait, yes - you are right. We are in Germany, so our friends will take the German solution, of course.That means: Voting!

Problem

As you can imagine, the friends do not just vote for a certain place to go to depending on their current mood.It really depends much more on who is there on a given Wednesday. Here are the rules:

1. Anne will always vote for going to the cinema.
2. Bob will vote for going to the disco if Karin is there, too.
3. Karin will vote for going to the disco if Charly is there, if Charly is not there but Anne is, Karin will vote for going to the cinema, if Charly and Anne are not there Karin will vote for going to the cocktail bar.
4. Dave is one of those guys who never votes for anything.
5. If Dave is there, Bob will vote for going to the cocktail bar.
6. If Charly and Anne are there, both of them will vote for going to the cinema.
7. Edward votes for going to the cocktail bar if Anne is there and Charly is not there, otherwise Edward votes for going to the cinema.
8. If Edward is there, Bob will vote for going to the cocktail bar.
9. Frank will vote for going to the cinema if neither Bob nor Anne are there, and Frank will vote for going to the disco if Anne is there.
10. If Anne is not there, Bob will vote for going to the cocktail bar.

For a given Wednesday, count the votes for each place from the friends who are there on that Wednesday.If no rule applies for a present friend or if several rules apply and require the friend to vote for different places, this friend abstains from voting on that Wednesday.

Input

The first line contains the number of scenarios (Wednesday meetings).

For each scenario, there is one line containing the names of the present friends. Names are separated by single blanks and consist of letters only. The same name will not occur more than once on the same line.

Output

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1.

Then for each scenario print a single line containing the place that received the most votes. Possible places are 'cinema', 'cocktail bar', and 'disco'. If none of those places receives more votes than the other two print 'stay at the Hacienda'.

Terminate the output for the scenario with a blank line.

Sample Input

```
2
Frank Dave
Karin Frank
```

Sample Output

```
Scenario #1:
cinema

Scenario #2:
stay at the Hacienda
```

Manhattan 2025

Time Limit:1000MS Memory Limit:30000K

Description

Background

Manhattan in the year 2025 - it is so densely populated that its old two-dimensional grid of streets and avenues fails to provide enough space for all the traditional vehicles such as cars, bicycles, or busses.Accordingly, the newly developed 3D-Skyjetters become very popular, because they allow to pass the traffic jams on the ground by flying in the air. After a series of horrible accidents caused by 3D-Skyjetters cutting a corner, New York authorities have put into place new regulations of air traffic and are determined to enforce them rigorously. The key point of these regulations is that 3D-Skyjetters must follow virtual airways on a three-dimensional rectangular grid, easy enough for the New Yorkers who had to use the two-dimensional rectangular grid of roads on the ground all their life.

Problem

You own a company that rents out 3D-Skyjetters. As 3D-Skyjetters are in such an early state of development,they are far from being economical. So your customers keep running out of petrol at all the wrong places,and you need a system to inform them about their current range at all times.

You may assume that travelling from one intersection to the next in the grid takes one unit of petrol, no matter if the customer is moving horizontally or vertically, up or down. You may also assume that your customer is located at some intersection where his or her movements are not restricted by the ground or other obstacles, but just by the amount of remaining petrol.

Given the amount of petrol, provide a graphical representation of all the intersections in the range of your customer, along with the amount of petrol that is needed to go there.

Input

The first line of the input contains the number of scenarios. For each scenario, there is a line containing the units of remaining petrol, i.e an integer u satisfying 0 <= u <= 9. If more than 9 units of petrol remain, the customer will ignore the display anyway.

Output

Start the output for each scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a graphical (or rather textual) representation of all intersections that can be reached with the given amount of petrol, along with the units of petrol necessary to go there. In this graphical representation, print the slices of the smallest axis-aligned three-dimensional cube containing all the intersections in the range, and label the slices from the bottom to the top starting at 1. For each slice,start the output with a line containing "slice #s:", where s is the number of the slice. In the lines that follow, print a graphical representation of all the intersections in that slice, using

- the digits 0 to 9 for intersections in the range, representing the amount of petrol necessary to go there,
- and the dot "." for intersections not in the range.

Print an additional blank line after each scenario.

Sample Input

```
2
0
2
```

Sample Output

```
Scenario #1:
slice #1:
0

Scenario #2:
slice #1:
.....
.....
..2..
.....
.....
slice #2:
.....
..2..
```

.212.
..2..
.....
slice #3:
..2..
.212.
21012
.212.
..2..
slice #4:
.....
..2..
.212.
..2..
.....
slice #5:
.....
.....
..2..
.....
.....

Mayan Dates

Time Limit:1000MS Memory Limit:30000K

Description

Background

The NASA has secretly built a Time machine which allows them to travel into the past. Its first application is to explore the Mayan culture. They have been able to gather some important dates from Maya monuments which are suitable for further investigation. They are now faced with a problem - they need to convert those Mayan dates to the Gregorian format, which is the only kind of input their time machine understands.

Problem

Write a conversion program to calculate the Gregorian date, the Mayan longcount and the Mayan ritual date for a given Gregorian date or Mayan longcount.

Mayan Longcount

The starting date, day zero, for the Mayan calendar is the 13th of August 3114 BC (before Christ). From this day zero onwards the days to a certain date are counted. One day is called K'in, 20 K'in are a Winal.18 Winal are a Tun, 20 Tun make a K'atun and 20 K'atun will form a Bak'tun. Example: 12.4.9.13.3 tells us that 12 Bak'tun, 4 K'atun, 9 Tun, 13 Winal and 3 K'in (days) have passed since August 13th 3114 BC.

Mayan Ritual Dates

The Mayans had the solar year with 365 days, there were no leap days to correct the calendar. The year was split in 18 month with 20 days and one month with 5 days. The names of the months are the following:Pob,Wo, Sip, Sotz', Tzek, Xul, Yaxk'in, Mol, Ch'en, Yax, Sak, Keh, Mak, K'ank'in, Muwan, Pax, K'ayab,Kumk'u, Wayeb. The days of the month were numbered 1 to 19 (or 1 to 4), the last day of the month was noted as day zero of the following month (i.e. you would not write 20 Pop, but 0 Wo to note the last day of the first month, the same applies to the five day month called Wayeb, its last day 5 Wayeb is written 0 Pop).Additionally to the months and days notation there is a notation which could be compared to our weeks and weekdays. Every day is labelled by both a number in the range from 1 to 13 and one of the following 20 names: Imix, Ik', Ak'bal, K'an, Chikchan, Kimi, Manik', Lamat, Muluk,

Ok, Chuwen, Eb, Ben, Ix, Men,Kib, Kaban, Etz'nab, Kawak, Ahaw. Both the numbers and the names are running in cycles, so the count will go: 1 Imix, 2 Ik', 3 Ak'bal, ... 13 Ben, 1 Ix, 2 Men. It will thus take a 260 day cycle (called the ritual year) until 1 Imix is counted again (compare Figure 1 on next page).
The weekday and month notation are used together as in the following example: 2 Kimi (weekday) 10 Yax (day and month). The 13th of August 3114 BC has the ritual date 4 Ahaw 8 Kumk'u.

Gregorian Dates
In our calendar we use leap days. A leap day is inserted every 4 years, except every 100 years. Overruling this a leap day will be inserted every 400 years.
It is also important to know that there is no year 0. Accordingly, 31th of December 1 BC is followed by 1st of January AD 1. The last leap year before Christ is 1 BC (the 400 year rule applies here), the next leap year after that is AD 4.

Input

The first line contains the number of scenarios.
For each scenario, there is one line containing either a Mayan longcount or a Gregorian date in the following formats:

- A Mayan longcount consists of 5 non-negative integers separated by dots.
- A Gregorian date consists first of three integers d, m, and y, separated by slashes, and satisfying 1 <= d <= 31, 1 <= m <= 12, and 0 < y. The values may or may not be zero padded, but will never exceed two characters for d and m, and four for y. This representation is always followed by a single blank and either "BC" or "AD".

We will neither ask for dates preceding August 13th 3114 BC, nor for dates after 13.0.0.0.0 which is the last day in the Mayan longcount calendar. Also, you will never be given invalid dates in either calendar.

Output

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then, regardless of the format given in the input, three lines have to be printed representing the given date in the following order:
1. the Gregorian date in the same format as specified in the input description, but without any padding zeros;
2. the Mayan longcount representation of the date, also as described before;

3. the Mayan ritual date consisting of four parts separated by single blanks: the weekday notation with its number and name, then the day of the month followed by the name of the month.

After these three lines, always a blank line must be printed.

Sample Input

2

13/08/3114 BC

12.19.10.6.14

Sample Output

Scenario #1:

13/8/3114 BC

0.0.0.0.0

4 Ahaw 8 Kumk'u

Scenario #2:

28/6/2003 AD

12.19.10.6.14

9 Ix 2 Tzek

Hint

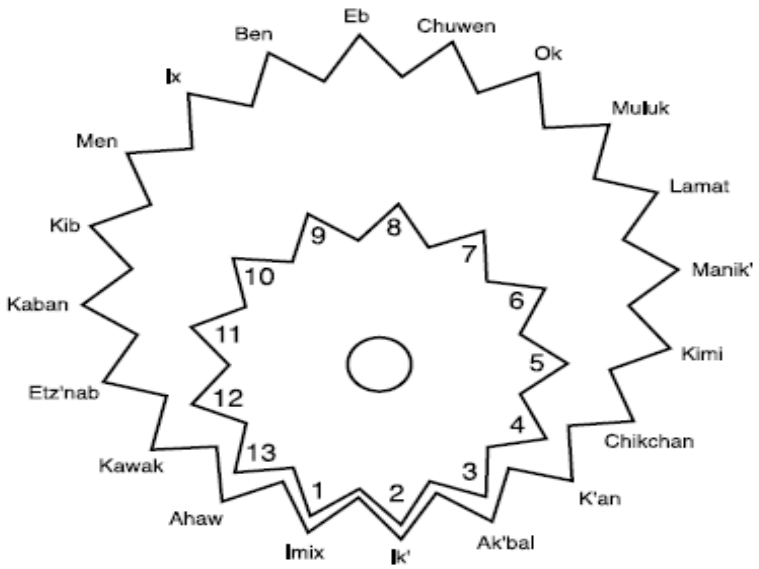


Figure 1: Mayan Ritual Dates

Quadratic Residues

Time Limit:1000MS Memory Limit:30000K

Description

Background

In 1801, Carl Friedrich Gauss (1777-1855) published his "Disquisitiones Arithmeticae", which basically created modern number theory and is still being sold today. One of the many topics treated in his book was the problem of quadratic residues.

Consider a prime number p and an integer a !≡ 0 (mod p). Then a is called a quadratic residue mod p if there is an integer x such that

$x^2 \equiv a \pmod{p}$,

and a quadratic non residue otherwise. Lagrange (1752-1833) introduced the following notation, called the "Legendre symbol":

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a \text{ is a quadratic residue mod } p \\ -1, & \text{if } a \text{ is a quadratic non residue mod } p \end{cases}$$

For the calculation of these symbol there are the following rules, valid only for distinct odd prime numbers p, q and integers a, b not divisible by p:

1. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$
2. $\left(\frac{1}{p}\right) = 1$
3. $a \equiv b \pmod{p} \quad \Rightarrow \quad \left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$
4. $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}, \left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$
5. $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}$

The statements 1. to 3. are obvious from the definition, 4. is called the Completion Theorem, and 5. is the famous Law of Quadratic Reciprocity for which Gauss himself gave no less than six different proofs in the "Disquisitiones Arithmeticae". Knowing these facts, one can calculate all possible Legendre symbols as in the following example:

$$\left(\frac{29}{79}\right) \overset{5.}{=} (-1)^{78\cdot 28/4}\left(\frac{79}{29}\right) = \left(\frac{79}{29}\right) \overset{3.}{=} \left(\frac{-8}{29}\right) \overset{1.}{=} \left(\frac{-1}{29}\right)\left(\frac{2}{29}\right)^3 = \left(\frac{-1}{29}\right)\left(\frac{2}{29}\right)$$
$$\overset{4.}{=} (-1)^{28/2}(-1)^{(29^2-1)/8} = (-1)^{14}(-1)^{105} = -1$$

Input

The first line contains the number of scenarios.

For each scenario, there is one line containing the integers a and p separated by a single blank, where $2 < p < 10^9$ is an odd prime, and a satisfies both $a \not\equiv 0 \pmod p$ and $|a| <= 10^9$.

Output

Start the output for every scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing (a/p), followed by a blank line.

Sample Input

```
3
29 79
2 29
1 3
```

Sample Output

```
Scenario #1:
-1

Scenario #2:
-1

Scenario #3:
1
```

Regetni

Time Limit:3000MS Memory Limit:30000K

Description

Background

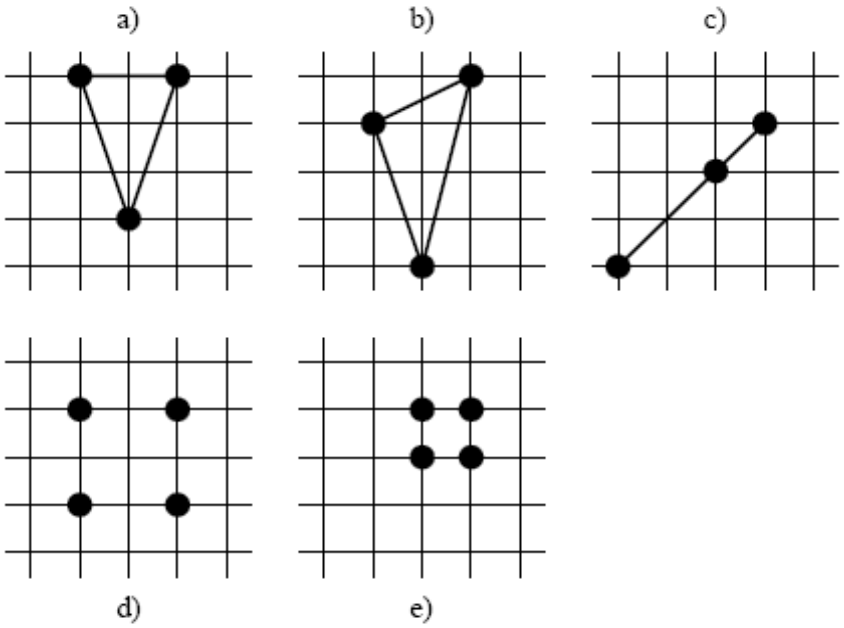
Hello Earthling. We're from the planet Regetni and need your help to make lots of money. Maybe we'll even give you some of it.

You see, the problem is that in our world, everything is about integers. It's even enforced by law. No other numbers are allowed for anything. That said, it shouldn't surprise you that we use integer coordinate systems to plan our cities. So far only axis-aligned rectangular plots of land have been sold, but our professor Elgnairt recently had the revolutionary idea to sell triangular plots, too. We believe that the high society will love this concept and it'll make us rich.

Unfortunately the professor patented his idea and thus we can't just do it. We need his permission and since he's a true scientist, he won't give it to us before we solve some damn riddle. Here's where you come in,because we heard that you're a genius.

Problem

The professor's riddle goes like this: Given some possible corners for the triangles, determine how many triangles with integral size can be built with them. Degenerated triangles with empty area (i.e. lines) have to be counted, too, since 0 is an integer. To be more precise, count the number of triangles which have as corners three different points from the input set of points. All points in a scenario will be distinct, i.e. there won't be duplicates. Here are some examples:



Example a) shows a triangle with integral area (namely 3), b) shows one with non-integral size, c) shows a degenerated triangle with empty area (i.e. zero, so count it!), d) shows four points of which you can choose any three to build an integral area triangle and e) shows four points where you can't build any integral area triangles at all.

Hint: The area A of a triangle with corners (x_1, y_1), (x_2, y_2) and (x_3, y_3) can be computed like this:

$A=|x_1y_2 - y_1x_2 + x_2y_3 - y_2x_3 + x_3y_1 - y_3x_1|/2$

Try to make clever use of this formula.

Input

The first line contains the number of scenarios. For each scenario, there is one line containing first the number N of distinct points in that scenario (0 <= N <= 10000) and after that N pairs of integers, each pair describing one point (xi, yi) with -100000 <= xi, yi <= 100000. All these numbers are separated by single blanks.

Output

Start the output for every scenario with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing the number of triangles with integral area whose three distinct corners are among the points given. Terminate the output for each scenario with a blank line.

Sample Input

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

3 0 0 2 2 3 3

4 0 0 2 0 0 2 2 2

4 0 0 1 0 0 1 1 1

9 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2

Sample Output

Scenario #1:

1

Scenario #2:

0

Scenario #3:

1

Scenario #4:

4

Scenario #5:

0

Scenario #6:

48