Problem A: Alphacode

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

> Alice: "Let's just use a very simple code: We'll assign 'A' the code word 1, 'B' will be 2, and so on down to 'Z' being assigned 26."
> Bob: "That's a stupid code, Alice. Suppose I send you the word 'BEAN' encoded as 25114. You could decode that in many different ways!"
> Alice: "Sure you could, but what words would you get? Other than 'BEAN', you'd get 'BEAAD', 'YAAD', 'YAN', 'YKD' and 'BEKD'. I think you would be able to figure out the correct decoding. And why would you send me the word 'BEAN' anyway?"
> Bob: "OK, maybe that's a bad example, but I bet you that if you got a string of length 500 there would be tons of different decodings and with that many you would find at least two different ones that would make sense."
> Alice: "How many different decodings?"
> Bob: "Jillions!"

For some reason, Alice is still unconvinced by Bob's argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

Input

Input will consist of multiple input sets. Each set will consist of a single line of digits representing a valid encryption (for example, no line will begin with a 0). There will be no spaces between the digits. An input line of '0' will terminate the input and should not be processed

Output

For each input set, output the number of possible decodings for the input string. All answers will be within the range of a **long** variable.

Sample Input

```
25114
1111111111
3333333333
0
```

Sample Output

```
6
89
1
```

B. Perilous Pond

Description

With a tap of her wand and a wave of her broom, the wicked witch turned both the prince and the princess into frogs and set them free to enjoy life as commoners. Shortly after the formerly royal lovers began exploring their new saliental existence, they gained a new perspective on the dangers of the park. Their old friend the happy-go-lucky duck now seemed to consider them a dinner second only to breadcrumbs.

The former princess hopped some number of lilies ahead of her lover in a game of hard to get – only to discover that she had been cornered by a duck who was clearly eyeing her as dinner. Her only hope is that her prince will grab a mouthful of nearby breadcrumbs and deliver them to the duck, before she becomes the main course.

The breadcrumbs are at hand, or rather in mouth, but frog navigation isn't trivial – especially for those newly frogged. Navigating a pond requires leaping from lily pad to lily pad. And, lily pads can be delicate platforms. If a frog leaps too far, the pad can collapse upon take-off or landing. And, unfortunately, the witch wasn't kind enough to transform these royals into frogs that can swim.

But, frog instincts are good. A frog can rapidly survey a pond, locate the lily pads, and determine the greatest leap that each will support. Since a frog must both jump and land, and the force of each is equal, a frog can jump no more than is supported by the weaker of the two pads end points. When jumping, frogs can move at a rate of one foot per second.

Your job is to determine if the prince can save his princess.

Input

The first line of the input indicates the number of games described. Each subsequent game description contains several lines describing the game.

The pond is rectangular. The first line of each game's description specifies the dimensions of the pond, in feet and fractions thereof. It contains two numbers, the width of the pond and the length of the pond.

The next line is a single integer, L, $2 <= L <= 1000$ which is the number of lily pads in the game. The next L lines specify the locations of the lily pads, one pad per line. The locations are measured from the corner which, from the prince's perspective, is on the far left. The first number per line is the distance to the right, a.ka., the x-coordinate. The second number per line is the distance from the back of the pond, a.k.a, the y-coordinate. The third number in the line is the strength of the pad – in other words, the maximum length of a leap that can be made to or from it.

Appearing after each of the three numbers on each line is a character, one of *S*, *F*, or *I*. *S* signifies the starting location, i.e., the initial location of the prince. *F* signifies the finish location, i.e., the location of the princess and the duck. *I* signifies any other, possibly intermediate, pad.

The last line of each game's description contains a single number, the amount of time, in seconds and fractions thereof, until the princess becomes dinner.

Output

There should be one line of output for each of the games in the input. The output should be presented in the same order as the input.

In the event that the princess becomes dinner, the output should consist of a single line, DINNER! In the event that the princess is saved, the output should consist of two numbers separated by a single space. The first number should be the total number of hops. The second number should be the amount of time the prince had to spare.

Sample Input

```
2
6.5 6.5
8
3.5 0.0 2.25 F
3.5 2.0 2.25 I
3.5 4.0 2.25 I
3.5 6.0 2.25 S
2.5 3.0 5.25 I
5.5 2.0 3.0 I
5.5 3.0 9.0 I
5.5 4.0 6.2 I
7.5
9.5 7.2
5
1.5 3.5 3.9 S
3.5 1.5 3.0 I
2.5 1.5 3.0 I
3.5 3.5 1.0 I
4.5 3.5 1.0 F
99.5
```

Sample Output

```
3 1.5
DINNER!
```

C. Counting Triangles

Description

Young Daphne lay in bed one night staring out the window, looking up at the wondrous sky. A sky full of stars. When she was little, she fell asleep by counting the stars. But now, in her adolescence, that game seems too easy. Instead, tonight, she has decided to count triangles. She wants to know how many unique triangles she can form from the stars in the sky – and resolves that she won't sleep until this question is answered. She considers two triangles to be unique if and only if they differ in the length of at least one side.

Fortunately for Daphne and her early morning alchemy class, she has already cataloged each of the stars visible from her window and assigned them coordinates in the Cartesian plane, such that the upper left corner of her view has the coordinates (0,0).

Your job is to help Daphne by writing a program that will, given the stars' coordinates as input, compute the total number of unique triangles.

Input

The input consists of *V* views, where each view consists of a list of *N* stars. The first line of input is the number of views, *V*. Subsequent input describes each of these *V* views.

Each view's description begins with a single line containing a non-negative integer, *N*, $1 <= N <= 500$ the number of stars visible in this view. Each subsequent line within a view's description contains two non-negative integers, $X <= 10000$, and $Y <= 10000$, the (X,Y) coordinates of a star as visible from this view.

Output

For each of the *N* views within the input, there should be one line of output. This line should contain the number of unique triangles within the view. The views should be represented in the output in the same order as they appear within the input.

Sample Input

```
3
5
0 0
1 0
1 1
0 2
2 1
5
0 0
0 1
0 2
1 1
3 1
6
5 5
```

```
7 5
5 7
6 10
8 10
6 12
```

Sample Output

```
6
5
12
```

D. Lawmakers

The leader of a small country is responsible for approving or vetoing a list of proposed new laws. There are *N* laws, numbered 1 through *N*, with *N* no more than 10,000. He has assembled a board of advisors containing the nation's top experts in a wide array of fields to help with the decision. The advisors have been asked to make recommendations. A simple recommendation can come in either of two forms:

Accept X
Reject X

The first indicates that the good of the country depends on accepting law X and the second indicates that the good of the country depends on rejecting law X. The leader also allows his advisors to submit compound recommendations of the form:

[simple recommendation] or [simple recommendation]

A compound recommendation indicates that, for the good of the country, at least one of the two simple recommendations must be followed.

The leader would like to comply with all the recommendations, but sometimes that turns out to be impossible! When that happens the leader's secretary must schedule a meeting among the advisors to rethink their recommendations.

You, the leader's secretary, are responsible for determining whether complying with all the recommendations is impossible.

Input format:

The first line is a single integer *C*, the number of countries described. Each country description contains the following:

N [Number of simple recommendations] [Number of compound recommendations]
[simple recommendations, 1 per line]
[compound recommendations, 1 per line]

Output format:

One line consisting of the text "Meeting required" or "Meeting not required"

Sample Input:

```
1
2 2 1
Accept 1
Reject 2
Accept 1 or Reject 2
```

Sample Output:

```
Meeting not required
```

E. Mosquito Planet

Description

After Darth Vader had attacked the Rebel base on the ice planet of Hoth, the Rebels temporarily moved to another unexplored planet, and set their base on a huge open plateau. The planet's dominant life form was mosquitos, and the rebels had to deploy a huge number of Counter-Mosquito Units (CMUs). These units were simple robots, which walked to specified locations on the plateau and then continuously sprayed mosquito repellent; once a CMU began spraying, it could no longer change its location. The Rebels later discovered that the winds on the plateau always blew to the north and east, which affected the distribution of repellent. Specifically, if a CMU's coordinates were (x_0, y_0), then its repellent covered all points to the north and east of (x_0, y_0), as shown in Figure 1; in other words, it covered all points (x, y) such that $x >= x_0$ and $y >= y_0$. After the Rebels had noticed it, they realized that some CMUs were redundant; that is, they could be removed without reducing the protected area. For example, if the deployment included four CMUs in Figure 2, then the CMUs (2, 3) and (4, 4) were redundant. To save repellent, the Rebels decided to remove the redundant units. Your task is to write a program that determines the number of these redundant units.

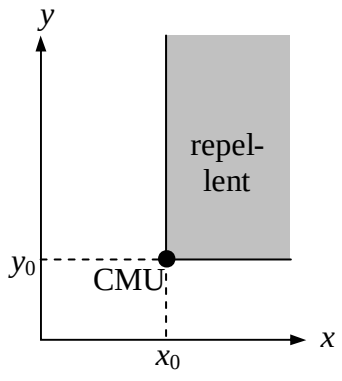

Figure 1

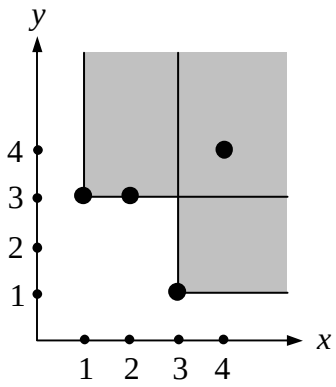

Figure 2

Input

The input includes multiple test cases; the first line of the input is a single integer between 1 and 100, which is the number of test cases. The first line of each test case is a single integer between 1 and 1000000, which is the number of counter-mosquito units. The other lines of the test case represent these units, one unit per line. The description of a unit consists of two integers between 1 and 1000000, separated by a single space, which are the unit's coordinates.

Output

The output should show the number of redundant CMUs for each test case; each number should be on a separate line, with no surrounding spaces.

Sample Input

```
3
1
1 1
2
1 1
2 2
4
1 3
2 3
3 1
4 4
```

Sample Output

```
0
1
2
```

Problem F: Team Rankings

It's preseason and the local newspaper wants to publish a preseason ranking of the teams in the local amateur basketball league. The teams are the Ants, the Buckets, the Cats, the Dribblers, and the Elephants. When Scoop McGee, sports editor of the paper, gets the rankings from the selected local experts down at the hardware store, he's dismayed to find that there doesn't appear to be total agreement and so he's wondering what ranking to publish that would most accurately reflect the rankings he got from the experts. He's found that finding the *median ranking* from among all possible rankings is one way to go.

The median ranking is computed as follows: Given any two rankings, for instance **ACDBE** and **ABCDE**, the *distance* between the two rankings is defined as the total number of pairs of teams that are given different relative orderings. In our example, the pair **B**, **C** is given a different ordering by the two rankings. (The first ranking has **C** above **B** while the second ranking has the opposite.) The only other pair that the two rankings disagree on is **B**, **D**; thus, the distance between these two rankings is 2. The median ranking of a set of rankings is that ranking whose sum of distances to all the given rankings is minimal. (Note we could have more than one median ranking.) The median ranking may or may not be one of the given rankings.

Suppose there are 4 voters that have given the rankings: **ABDCE**, **BACDE**, **ABCED** and **ACBDE**. Consider two candidate median rankings **ABCDE** and **CDEAB**. The sum of distances from the ranking **ABCDE** to the four voted rankings is $1 + 1 + 1 + 1 = 4$. We'll call this sum the *value* of the ranking **ABCDE**. The value of the ranking **CDEAB** is $7 + 7 + 7 + 5 = 26$.

It turns out that **ABCDE** is in fact the median ranking with a value of 4.

Input

There will be multiple input sets. Input for each set is a positive integer n on a line by itself, followed by n lines (n no more than 100), each containing a permutation of the letters **A**, **B**, **C**, **D** and **E**, left-justified with no spaces. The final input set is followed by a line containing a 0, indicating end of input.

Output

Output for each input set should be one line of the form:

ranking `is the median ranking with value` *value*.

Of course *ranking* should be replaced by the correct ranking and *value* with the correct value. If there is more than one median ranking, you should output the one which comes first alphabetically.

Sample Input

```
4
ABDCE
BACDE
ABCED
ACBDE
0
```

Sample Output

```
ABCDE is the median ranking with value 4.
```

Problem G: To and Fro

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is "There's no place like home on a snowy night" and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character 'x' to pad the message out to make a rectangle, although he could have used any letter.

Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2 ... 20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Sample Input

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Sample Output

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

H. Prime Gambling

Description

The university administration has decided to install several slot machines in the university center, which are called Casino Machines for Universities (CMUs). This machines are similar to the usual casino slot machines, but the gambling is based on various science facts. In particular, one machine allows students to bet on the number of primes in a specified interval. After a student drops a quarter into this machine, it displays an interval of integers, and the student has to guess the number of primes in this interval. If her guess is larger than the correct answer, she loses; if her guess is an underestimate, she gets back her quarter; finally, if her guess is right, she wins a T-shirt with the slogan "Gamble less, study more." Your task is to write a program that verifies the correctness of guesses.

Input

The input is a list of games, one game per line. The first line of indput indicates the total number of games. Each subsequent line describes a single game. The description of a game includes three integers, separated by single spaces. The first two integers are the beginning and end of an interval; these integers are between 2 and 1,000,000, and the first of them is no greater than the second. The third integer is a student's guess about the number of primes in this interval, which is between 0 and 100,000. The total number of games is at most 1,000,000.

Output

The output shows the outcome of each game, one outcome per line. If the guess is greater than the correct answer, the outcome is "*loss*"; if it is less than the correct answer, the outcome is "*draw*"; finally, if it is correct, the outcome is "*win*".

Sample input

```
4
2 2 1
4 4 0
2 5 2
2 6 4
```

Sample output

```
win
win
draw
loss
```