Problem S: Say it loudly!

Alice was walking down the street. When she saw Bob on the other side, she shouted: "`HELLO, BOB!`"

"`Hi, Alice!`" responded Bob from afar and they both hurried to meet in the middle of the street.

"`Bob! Why didn't you shout back when I said hello?`" asked Alice, apparently a bit annoyed.

Bob, surprised, defended himself: "`What? But I did. I said 'Hi, Alice!'`".

Alice smiled. "`Oh, I see. You were too quiet. To shout, you must speak in uppercase!`"

"`Ah, OK. No, wait. I said 'Hi', which has an uppercase letter, see?`"

"`That makes no difference. The whole word must be in uppercase, LIKE THIS. Words that are entirely in uppercase are twice as loud as normal words.`"

"`Your shouting is only twice as loud as normal?`" laughed Bob. "`That's not very much.`"

"`Just you wait!`" responded Alice and produced a terrible high-pitched squeal: "`*Eeeeeeeee!*`"

"`How did you do that?`"

"`You just have to add asterisks. All words between them are three times louder.`"

"`Cool. Let's see who can shout louder!`" proposed Bob and screamed: "`*AAA aaa* a a aargh!`"

Alice countered: "`*aaaaa*!`"

"`I won!`" exclaimed Bob. "`I used three normal words, one word that was thrice as loud, and one that was six times as loud. You could say I scored 1+1+1+3+6=12 loudness points.`"

"`No, you've got it all wrong. You have to count the average, not the sum. So you only scored 2.4 points. And I scored 3, so I won!`" concluded Alice.

You are given several sentences. Your task is to find the loudest one.

Input and output specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

The first line of each test case contains an integer n ($1 \leq n \leq 100$), the number of sentences. Each of the following n lines contains one sentence. Each line is at most 500 characters long.

A *word* is a maximal non-empty sequence of consecutive upper- and lowercase letters of the English alphabet. Words in a sentence are separated by spaces ( ), punctuation (`,.:;'"!?-`) and asterisks (`*`). No other characters appear in the input. Each sentence contains at least one word. There is an even number of asterisks in each sentence. The input file for the **easy subproblem S1** contains no asterisks.

For each test case, output one line with a single integer – the index of the loudest sentence (i.e., the one whose average word loudness is the highest), counting from 1. If there are multiple sentences tied for being the loudest, output the smallest of their indices.

Example

input

```
2

2
HELLO, BOB!
Hi, Alice!

4
*AAA aaa* a a aargh!
*aaaaa*
*note*that asterisks do not*nest***
* * THIS IS NOT BETWEEN ASTERISKS * *
```

output

```
1
2
```


Problem T: Two covers

In a typical genome assembly problem, we are given set of small strings called *pieces* and our task is to find their superstring with some reasonable properties. In this problem, you are given one such superstring and a collection of pieces aligned to that superstring. Your task is to evaluate one particular property.

Problem specification

You are given the length of the superstring ℓ, a list of n pieces, and a number k. The positions in the superstring are numbered from 1 to ℓ. For each piece i you are given the positions (b_i, e_i) of its beginning and end.

The letter at position x in the superstring is *well-covered* if:

- There is a piece (b_i, e_i) such that $b_i \leq x - k$ and $x \leq e_i$.
- There is a **different** piece (b_j, e_j) such that $b_j \leq x$ and $x + k \leq e_j$.

Your task is to count the number of letters which are **not** well-covered.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with a line containing three integers: ℓ, n, and k. Each of the following n lines contains two integers b_i and e_i ($1 \leq b_i \leq e_i \leq \ell$): the indices of the endpoints of a piece. The pairs (b_i, e_i) are all distinct.

In the **easy subproblem T1** you may assume that $t \leq 20$, $\ell \leq 10\,000$, $n \leq 1000$, and $1 \leq k \leq n$.

In the **hard subproblem T2** you may assume that $t \leq 10$, $\ell \leq 10^{17}$, $n \leq 1\,000\,000$, and $1 \leq k \leq n$.

Output specification

For each test case, output a single line with a single integer – the number of letters that are not well-covered.

Example

input	output
1 8 3 2 1 4 3 5 4 8	5

The well-covered letters are at positions 3, 4, and 5. Note that the letter at position 6 is not well-covered.

Problem U: Urban planning

The town of Pezinok wants to expand by building a new neighborhood. They hired some famous architects to design it, and your friend Jano is one of them. He is in charge of the road network. It is common to make one-way roads these days, so Jano went all out and decided to make *all* the roads one-way. (Of course, a pair of junctions can be connected by two roads – one in each direction.)

Once Jano made a map showing the planned roads, he noticed that some parts of the neighborhood might not be reachable from other parts. To estimate the impact of this issue, he decided to use a systematic approach: he took a piece of paper and wrote everything down. Namely, for each junction j he listed all other junctions that are (directly or indirectly) reachable from j. We call this information the *reachability list* of a road network.

But then Jano's hard drive crashed and he lost all the plans he had made. The only thing he has left is the piece of paper with the reachability list.

Help Jano reconstruct his original road network. Of course, many different road networks can produce the same reachability list. Therefore, Jano asked you to find the smallest possible road network that has the given reachability list. That should help him reconstruct his original plans.

Problem specification

Find a road network with the smallest possible number of roads that has the given reachability list.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with a line containing an integer n, denoting the number of junctions. The junctions are numbered 1 through n. Next, n lines follow, each containing a string of length n. The i-th of these lines specifies which junctions are reachable from junction i. Namely, the j-th character in the line is 1 if junction j is reachable from i and 0 otherwise. (Note that for each i, junction i is reachable from itself.)

The reachability list is consistent – it describes at least one real road network.

In the **easy subproblem U1** you may further assume that the reachability list is special: for every pair of junctions $a \neq b$, either a is reachable from b or b is reachable from a, but never both.

Output specification

For each test case, output the smallest road network that corresponds to the given reachability list. The first line of the description should contain the number of roads m (which has to be as small as possible). Each of the next m lines should contain two integers a_i and b_i ($1 \leq a_i, b_i \leq n$) such that there is a one-way road going from junction a_i to junction b_i.

You can print the roads in any order. If there are multiple optimal solutions, output any of them.

Example

input

```
2

3
111
011
001

4
1111
1111
0011
0011
```

output

```
2
1 2
2 3

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

The first test case satisfies the additional constraint for the easy subproblem.

In the second test case, we know that all junctions should be interconnected, except that junctions 1 and 2 should not be reachable from junctions 3 and 4. The smallest road network with this property has five one-way roads. One such road network is shown in the example output above. This test case also has other optimal solutions.

