Army strength

The next MechaGodzilla invasion is on its way to Earth. And once again, Earth will be the battleground for an epic war.

MechaGodzilla's army consists of many nasty alien monsters, such as Space Godzilla, King Gidorah, and MechaGodzilla herself.

To stop them and defend Earth, Godzilla and her friends are preparing for the battle. (After this contest, you can visit `http://www.atari.com/godzilla/` to find out more about them.)

Problem specification

Each army consists of many different monsters. Each monster has a strength that can be described by a positive integer. (The larger the value, the stronger the monster.)

The war will consist of a series of battles. In each battle, the weakest of all the monsters that are still alive is killed.

If there are several weakest monsters, but all of them in the same army, one of them is killed at random. If both armies have at least one of the weakest monsters, a random weakest monster of MechaGodzilla's army is killed.

The war is over if in one of the armies all monsters are dead. The dead army lost, the other one won.

You are given the strengths of all the monsters. Find out who wins the war.

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 line containing two positive integers N_G and N_M – the number of monsters in Godzilla's and in MechaGodzilla's army. Two lines follow. The first one contains N_G positive integers – the strengths of the monsters in Godzilla's army. Similarly, the second one contains N_M positive integers – the strengths of the monsters in MechaGodzilla's army.

Output specification

For each test case, output a single line with a string that describes the outcome of the battle.

If it is sure that Godzilla's army wins, output the string "`Godzilla`".

If it is sure that MechaGodzilla's army wins, output the string "`MechaGodzilla`".

Otherwise, output the string "`uncertain`".

Example

input

```
2

1 1
1
1

3 2
1 3 2
5 5
```

output

```
Godzilla
MechaGodzilla
```

In the first test case, there are only two monsters, and they are equally strong. In this situation, MechaGodzilla's monster is killed and the war ends.

In the second test case, the war will consist of three battles, and in each of them one of Godzilla's monsters dies.

Breaking in

Mayco has recently been hired as a security consultant for a well-known software company. At the moment, he's working on his first assignment – trying to determine which of the company's servers would be the best targets for potential attackers. It is a bit difficult, though, because some of the servers "trust" some of the others. If an attacker compromises a server, he or she can also freely access all servers that trust it (and servers that trust them, and so on).

By definition, the importance of a server S is the number of servers the attacker would be able to access if he compromised S. The most important servers are those with the highest importance. (Note that there can be more than one most important server. This is also illustrated in the example below.)

Problem specification

The network consists of N computers, numbered 1 to N, inclusive. The trust between computers is described by M ordered pairs (A, B) of numbers, denoting that computer A trusts computer B. The trust is not assumed to be mutual – i.e., if a computer A trusts computer B, it does not necessarily imply that computer B trusts computer A.

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 the numbers N and M. Each of the following M lines contains two integers, A and B, denoting that computer A trusts computer B.

Output specification

For each test case, the output shall contain one line with the numbers of all of the most important servers. The numbers must be listed in increasing order and separated by single spaces.

Example

input

```
2

5 4
3 1
3 2
4 3
5 3

6 5
1 2
2 3
3 1
1 4
5 6
```

output

```
1 2
4
```

In the first test case, compromising one of the servers 1 and 2 gives us access to the entire network.

In the second test case, compromising server 4 gives us access to servers 1, 2, and 3. All other servers are less important.

Comparison mysteries

For beginners in programming it often comes as a great surprise when numbers inside a computer refuse to behave the same way numbers in mathematics do. In this task, we will investigate two such cases.

The ideas needed to solve this task are neither platform-specific, nor language-specific. However, due to many subtle differences between various programming languages the best way to state this task was to pick a single language. In this case that language will be Java.

We will be using numeric variables only, and only the following six basic types:

- `byte`, a signed integer from -128 to 127, inclusive
- `short`, a signed integer from -32768 to 32767, inclusive
- `int`, a signed integer from -2147483648 to 2147483647, inclusive
- `long`, a signed integer from -9223372036854775808 to 9223372036854775807, inclusive
- `float`, a 32-bit floating point number (24 bits sign+mantissa, 8 bits exponent)
- `double`, a 64-bit floating point number (53 bits sign+mantissa, 11 bits exponent)

For each of these types, the Scanner class has a corresponding method (e.g., `nextByte` for `byte`) that scans the next token of the input as the given type.

Easy data set

Obviously, in mathematics the only solution to $x = -x$ is zero. Now consider the following Java code:

Java code template

```
import java.util.*;
public class C1 {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in); sc.useLocale(Locale.US);
    ?????  x = sc.?????();
    System.out.println( (x==-x) + " " + (x!=0) );
  }
}
```

We would like you to show that there are cases when a non-zero number is equal to its own negation. Choose the type of the variable x and provide input to this program so that it outputs "`true true`".

Submit your answer as a text file with two lines. In the first line, write the type of the variable x, in the second line a string that shall be used as the input for your program.

An example submission follows. This submission is syntactically correct, but it does not produce the desired output.

Your submission

```
short
47
```

The corresponding Java code

```
import java.util.*;
public class C1 {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in); sc.useLocale(Locale.US);
    short x = sc.nextShort();
    System.out.println( (x==-x) + " " + (x!=0) );
  }
}
```

Its output on your input

```
false true
```

Hard data set

An even more obvious fact is that whenever $x = y$ and $y = z$, we must have $x = z$ as well. Now consider the following Java code:

Java code template

```
import java.util.*;
public class C2 {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in); sc.useLocale(Locale.US);
    ?????  x = sc.?????();
    ?????  y = sc.?????();
    ?????  z = sc.?????();
    System.out.println( (x==y) + " " + (y==z) + " " + (x==z) );
  }
}
```

We would like you to show that there are cases when equality is not transitive. Choose the types of the variables x, y, and z and provide input to this program so that it outputs "`true true false`".

Submit your answer as a text file with six lines. In the first three lines, write the types of the variables x, y, and z. In the second three lines write the strings that shall be used as the input for your program.

An example submission follows. This submission is syntactically correct, but it does not produce the desired output.

Your submission

```
short
short
int
47
47
47
```

The corresponding Java code

```
import java.util.*;
public class C2 {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in); sc.useLocale(Locale.US);
    short x = sc.nextShort();
    short y = sc.nextShort();
    int z = sc.nextInt();
    System.out.println( (x==y) + " " + (y==z) + " " + (x==z) );
  }
}
```

Its output on your input

```
true true true
```

Applets

For easy experimentation we provide an applet for each of the subproblems, where you can set variable types and values, and let the applet produce the output for you.

- Applet for the easy data set: `C1applet.html`
- Applet for the hard data set: `C2applet.html`

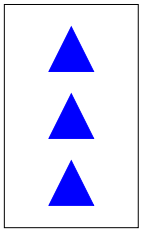
Java documentation

This section is not strictly necessary to solve the problem. But in case someone feels the need to browse the Java documentation to check something out, we provide links to the online version of *The Java Language Specification, Third Edition* you might consider useful:

- Main page:
  `http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html`
- 4.2 Primitive Types and Values:
  `http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html#4.2`
- 5.6 Numeric Promotions:
  `http://java.sun.com/docs/books/jls/third_edition/html/conversions.html#5.6`

Discover all Sets

The game of Sets is a very popular card game (`http://en.wikipedia.org/wiki/Set_(game)`). The game is played with a special deck of cards, where each card has a unique picture on it. This is an example of such a card:



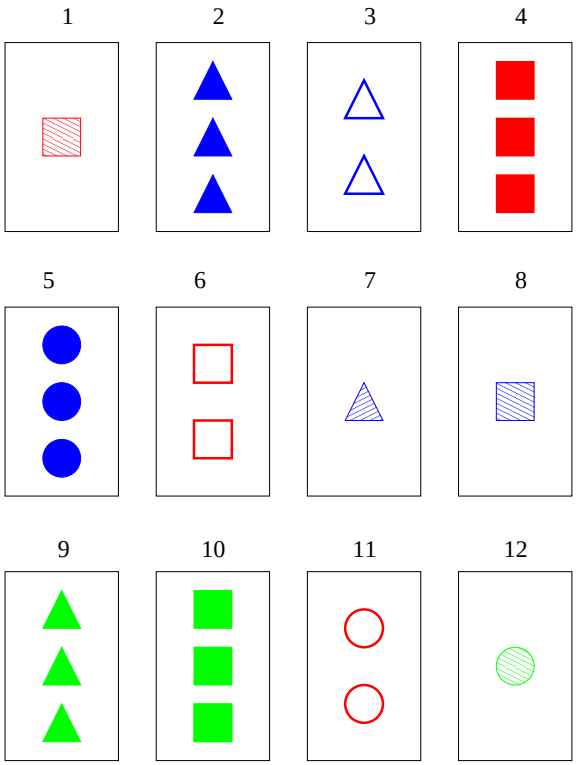
Each of the cards is uniquely identified by its four properties: the shape shown in the picture (square, triangle, circle), the count of these shapes (one, two, three), their color (red, green, blue), and the fill style (empty, striped or full).

For each combination of values there is exactly one such card in the deck. Thus the total number of cards in the deck of the original game is $3^4 = 81$.

The main concept of this card game is a **Set**. A Set consists of three cards such that for each property, they are either all the same, or all are different. In other words, three cards are **not** a Set if and only if you can say "Two of them are ... and one is not."

In the picture below, the cards 1, 4, and 6 form a Set: they are all red, all are squares, the counts are different, and so are the fill types.

The cards 2, 3, and 9 do not form a Set, for example because two of them are blue and one is not.



Problem specification

In this problem we will play a more general version of the game. There will be N card properties, and for each of these properties there will be M different values it can obtain. The deck will contain exactly one card for each possible combination of properties. Thus there will be M^N cards in the deck.

In this more general game, a Set consists of exactly M cards such that for **each** of the N properties the following holds: *Either all the cards have the same value of this property, or all of them have different values.*

You will be given the values N, M, a positive integer K, and a description of K different cards from the deck. Your task is to find the number of different Sets that can be created from the given cards. (Note that some of those Sets may overlap.)

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 looks as follows: The first line contains three positive integer N, M and K giving the number of different properties of the cards, the number of different values of a single property, and the number of cards drawn.

Each of the next K lines describes one of these cards. The i-th of these lines ($1 \leq i \leq K$) contains exactly N numbers $p_{i,j}$ ($1 \leq j \leq N$), $1 \leq p_{i,j} \leq M$. Value $p_{i,j}$ is the value of the j-th property of the i-the card.

Output specification

For each test case output a single line with a single integer – the number of different Sets that can be made from the given cards.

Example

input

```
1

4 3 12
1 1 1 2
2 3 3 3
2 2 3 1
1 3 1 3
3 3 3 3
1 2 1 1
2 1 3 2
1 1 3 2
2 3 2 3
1 3 2 3
3 2 1 1
3 1 2 2
```

output

```
11
```

This example input corresponds to the image in the problem statement. The four properties are described in this order: shape (square, triangle, circle), count (one, two, three pieces), color (red, green, blue) and fill style (empty, striped, full).

These are the 11 different sets: (1,4,6), (1,7,12), (2,3,7), (2,6,12), (3,4,12), (3,5,8), (4,5,9), (5,11,12), (6,8,10), (7,10,11), and (8,9,11).

Expected road cost

For many years, Absurdistan was famous for its camel caravans. However, the new Grand Vizier decided to make a radical step towards civilization – he will actually build some roads.

On the other hand, too many roads would unnecessarily confuse the local inhabitants. (That, and camels don't really like walking on roads.) Therefore the road network will, for now, have to be minimal, just enough to connect all the villages.

Preliminary investigations were made. For some pairs of villages we know a lower and an upper estimate of the cost of building a direct road between these two villages.

The Grand Vizier has made you personally responsible for preparing the budget. You know that after you prepare the budget, the exact cost for each potential road will be computed, and the best set of roads will be selected.

Problem specification

There are N villages in Absurdistan. There are M pairs of villages such that we can build a direct road that connects them. For each such road i, we know the minimal cost l_i and the maximal cost u_i.

The builders will first find the exact cost for each road, and then select the cheapest set of roads that connects all the villages.

Assuming that for each road its true cost is a real-valued random variable with uniform distribution over the interval $[l_i, u_i]$, compute the expected cost of the road network.

Notes

All roads are bidirectional.

The roads are built in such a way that they don't intersect anywhere, using bridges if necessary. Thus in order to connect N villages you have to build at least $N - 1$ roads.

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 consists of several lines. The first line of a test case contains two integers N and M, giving the number of villages and the number of possible roads, respectively. The villages are numbered from 0 to $N - 1$. M lines follow, each of them describes one possible road. Each description consists of four integers: the villages connected x_i and y_i, and the minimum and maximum cost l_i and u_i.

Villages are numbered starting from zero. All costs are non-negative. The input graphs are somewhat special, it may be worth your while to examine the input data before attempting to solve the task.

Output specification

For each test case, the output shall contain one line with the expected cost of the cheapest road network. This value is always rational, and you are to output it as the simplest possible fraction.

Print the fraction as A/B, with no spaces. Even if $B = 1$, print the denominator.

If for some test case it is not possible to build any road network that connects all villages, output -1 instead.

Examples

input

```
4

3 2
0 1 0 9
1 2 10 11

4 2
0 1 10 11
1 2 10 12

3 3
0 1 0 1
1 2 2 2
0 2 3 3

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

output

```
15/1
-1
5/2
3/4
```

In the first test case, we have to build both roads. Their expected costs are $9/2$ *and* $21/2$*, which gives a total of* $30/2 = 15/1$*.*

In the second test case, we can not connect village 4 to the other villages.

In the third test case, the two cheapest roads are always $1 - 2$ *and* $2 - 3$*, with total expected cost* $1/2 + 2 = 5/2$*.*

The fourth test case illustrates that the builders first find out the exact road costs, and only then pick the best set. In this case, we generate three random numbers from $\langle 0, 1 \rangle$*, and the result is the expected sum of the smaller two.*

Find the meaning

Do you remember the very-late-postcard delivery story from February this year? A Yellowstone postcard sent in 1929 was finally delivered this year, 79 years in transit. However, this postcard was not the only case; Mr. has received two letters dated 1929 this February as well. The fact that this didn't get into the news was that journalists couldn't find Mr. for an interview.

Each of the letters started with two handwritten sentences: *"Analyze this message to get the codeword. The codeword is a single word."* In each case, utter gibberish followed.

Problem specification

For each input file, you have to analyze it and find the codeword. The codeword is a single English word. (Did you expect anything else? `;-)`)

Output specification

For each test case, the output shall contain one line containing the codeword in **ALL UPPERCASE**. If you submit the correct output but not in ALL UPPERCASE, it will be considered incorrect.

Helpful advice

For the easy input, once you get past the first step, remember to look for the ANSWER.

For the hard input, the input file contains two copies of the same message, and the final answer is the name of an institution.

Example

input

```
The answer for this example is
Oscilloscope.  Yes, it is really
that simple.
```

output

```
OSCILLOSCOPE
```

Game on Words

Do you remember Jan from the last year's IPSC? He loves to play games with numbers. Now he is one year older and so he expanded his interests also to word games. Moreover he also involved his younger brother Maroš to play his new two player game. Poor small Maroš, he will never beat his older brother Jan without your help.

Problem specification

In the beginning, the players agree on a wordlist they'll use for the game. The words in the wordlist must consist of lowercase English letters only.

The game starts with an empty string S. Players take alternating turns, starting with Maroš. In each turn, the current player must append a letter to the end of S.

The player that either creates a string from the wordlist, or creates a string S that is not a prefix of any word in the wordlist, loses the game.

Your task is to play this game against our server, and win it.

Input specification

The input file contains the wordlist that will be used. Its first line contains one integer N – the number of words in the wordlist. N lines follow, each of them contains one of the words. No two words in the wordlist are equal, and all the words contain lowercase English letters only.

Output specification

Submit a text file with the description of the move you want to make.

If you want to continue playing the current game, the file shall contain a single line with a single lowercase letter – the letter you want to append to the current string.

If you want to give up the current game and start a new one, the file shall contain a single line of the form "`RESTART x`", where `x` is the lowercase letter you want to play in your first move.

Evaluation result

If your submission does not represent a valid move, you will get a corresponding error message, and the state of the game will not be altered.

If your submission causes you to lose the game, you will get a corresponding error message, and the game is restarted.

If your submission causes you to win the game, it will be accepted, and you will receive points for solving the corresponding input.

In the remaining case (a valid move that does not terminate the game), you will get a message describing our move.

Warnings

Note that for each data set there is a **strict limit**: you can only make a total of ten submissions. Play carefully – if you give up a game after 8 moves, chances are you will not have enough submissions left to win the second one.

Also, you may assume that our player knows the optimal strategy, and that it is really nasty `:-)`

Gameplay examples

Wordlist

```
5
cats
contest
dog
done
dragon
```

Communication 1

```
submit:  c
answer:  After your move:  'c'.  After my move:  'co'.
submit:  n
answer:  After your move:  'con'.  After my move:  'cont'.
submit:  e
answer:  After your move:  'conte'.  After my move:  'contes'.
submit:  t
answer:  You lost (created a banned word).  A new game starts.
```

Communication 2

```
submit:  c
answer:  After your move:  'c'.  After my move:  'co'.
submit:  x
answer:  You lost (nothing starts with 'cox').  A new game starts.
submit:  c
answer:  After your move:  'c'.  After my move:  'co'.
submit:  RESTART d
answer:  After your move:  'd'.  After my move:  'do'.
submit:  n
answer:  OK
```

Hidden text

A common practice when displaying a document that contains sensitive information is to post-process it to make the sensitive parts not readable, for example by blurring or pixelizing them.

However, in many situations this practice is not sufficient. An attacker might be able to restore the scrambled part of the text, at least partially. That will be your job in this problem.

Problem specification

You are given two image files as input. The easy input is a clear image with black letters and white background, the hard input contains various kinds of noise as well. In both cases, the image has been post-processed to hide a part of the information it contains. Recover it, and submit an appropriate proof.

Blurring algorithm used

The blurred part of the image was produced using the following algorithm:

Pick a positive integer σ. We define

$$G_{r,c} = e^{-(r^2+c^2)/(2\sigma^2)}$$

Compute the values $G_{r,c}$ for $-3\sigma \leq r, c \leq 3\sigma$. Let $S = \sum_{-3\sigma \leq r,c \leq 3\sigma} G_{r,c}$. Set $H_{r,c} = G_{r,c}/S$.

For example, for $\sigma = 1$ you should get the following values: $S = 6.279785$, and the values $H_{r,c}$ look as follows (with $H_{0,0}$ in the middle):

```
0.000020  0.000239  0.001073  0.001769  0.001073  0.000239  0.000020
0.000239  0.002917  0.013071  0.021551  0.013071  0.002917  0.000239
0.001073  0.013071  0.058582  0.096585  0.058582  0.013071  0.001073
0.001769  0.021551  0.096585  0.159241  0.096585  0.021551  0.001769
0.001073  0.013071  0.058582  0.096585  0.058582  0.013071  0.001073
0.000239  0.002917  0.013071  0.021551  0.013071  0.002917  0.000239
0.000020  0.000239  0.001073  0.001769  0.001073  0.000239  0.000020
```

To blur a greyscale image, first represent each pixel $[pr, pc]$ as an integer $I_{pr,pc}$ between 0 and 255, inclusive. Here, 0 is black, and 255 white. Pixels outside of the image are considered to be white. The color $B_{pr,pc}$ of a pixel $[pr, pc]$ after blurring can be computed as follows:

$$B_{pr,pc} = \sum_{-3\sigma \leq r,c \leq 3\sigma} H_{r,c} \cdot I_{pr+r,pc+c}$$

In words, the new color is a weighted average of the colors of the nearby pixels, and the weights used are the values $H_{r,c}$ defined above. The exact color $B_{pr,pc}$ is then rounded to the nearest integer from the set $\{0, \ldots, 255\}$.

For a true color image, blurring is done separately for its red, green, and blue color component.

For the easy data set we used the value $\sigma = 15$, for the hard data set $\sigma = 17$.

Input specification

The input is a Portable Network Graphics (PNG) file.

(See `http://en.wikipedia.org/wiki/Portable_Network_Graphics` if you are not familiar with this image format. All modern browsers and all modern image manipulation programs should be able to open PNG files.)

Output specification

For each input file, submit an output file that contains a single English word in **ALL UPPERCASE**. If you submit the correct output but not in ALL UPPERCASE, it will be considered incorrect.

The word you are supposed to output is uniquely specified by the hidden part of the input file. For the hard data set, the word has between 6 and 9 letters, and starts with the letter P.

Example

Input image

Original image without the distortion

Output file

```
BULL
```

Inventing Test Data

Preparing a problem set is a very hard task. There are always issues with clarity of problem statements, bugs in our solutions, input or output data, and so on. Sometimes, despite our best efforts, these issues are only found during the contest, and this can really spoil it.

To prevent this from happening in the future, we already started to prepare data for IPSC 2009, and we decided to use your help in doing so. Currently we are working on a simple textbook problem: "Given a weighted undirected complete graph, find its minimum spanning tree." (See the Definitions below if you are not sure what a spanning tree is.)

Almost everythig is already prepared for this problem: the problem statement, our solution, and also the desired output data. The only (and quite important) thing left is the input data. But creating it is not as simple as it looks.

The bad thing that can happen is that a graph can have more than one minimum spanning tree. If we used such a graph in the input data, we would have to write a complicated checker. And we are too lazy to do this. Therefore we want to find an input data that avoids such cases.

Moreover, we want the test data to be good. If all the other edges were much more expensive, the minimum spanning tree would be obvious, and many incorrect algorithms would be able to find it. Therefore we want all the edge weights to be as small as possible.

Definitions

A graph is a set of nodes, and a set of links. Each link connects two nodes. Each pair of nodes is connected by at most one link. Each link is assigned a positive integer (its weight). The sum of the weights of all links in a graph is the weight of that graph.

If every two nodes are connected by a link we say that the graph is complete.

A sequence of nodes $v_0, \ldots, v_n$ such that for each i the nodes v_i and v_{i+1} are connected by a link, is called a path.

If every two nodes in a graph are connected by a path, we say that the graph is connected.

If there is exactly one path between any two nodes we say that graph is a tree.

A spanning subgraph of a connected graph G is a connected graph that contains all nodes of G and some (not necessarily all) of its links.

A spanning subgraph T of a graph G is called the minimum spanning tree of G if and only if no other spanning subgraph has a smaller weight.

Note that a given graph can have more than one spanning tree. Also note that a spanning tree is always a tree.

Problem specification

Given a weighted tree T, you are to find the minimum possible weight of a complete graph G such that T is the only minimum spanning tree of G.

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.

First line of each test case contains an integer N – number of nodes in the tree. The nodes are numbered from 1 to N, inclusive. The following $N-1$ lines contain a description of the tree. Each of these lines contains three integers a_i, b_i, w_i meaning that node a_i is connected with node b_i by a link with weight w_i.

Output specification

For each test case, the output shall contain one line containing one integer – the minimum possible weight of a complete graph such that the given tree is its unique minimal spanning tree.

Example

input

```
2

3
1 2 4
2 3 7

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

output

```
19
12
```

In the first test case, we have to add a link between nodes 2 and 3 with weight at least 8.

In the second test case, the optimal graph contains the link $2-3$ with weight 2, and links $2-4$ and $3-4$ with weigths 3 each.

Just a single lie

Last week, Peter explained a simple game to his younger sister Alice: They agree upon a positive integer N. Peter will then pick a number from the set $\{0, 1, \ldots, N-1\}$ and Alice's goal is to guess Peter's number using the smallest number of questions. Alice is allowed to ask any question she wants, as long as it is a yes/no question about the number she tries to guess.

It did not take Alice long to discover the optimal strategy for this simple game. After she won a game with $N = 1024$ using just ten questions, the game started to be boring. But suddenly she got an idea how to make it more fun. With a smirk, she challenged Peter to be the one that will ask the questions.

Peter did not really care to play the game, so he just picked an arbitrary value L and asked: "Is your number less than L?" However, at that moment Alice announced the new twist she thought of: she may be lying in one of her answers. After this announcement, she answered Peter's first question.

Of course, this has caught Peter's attention, and now he wants to finish the game using the lowest possible number of questions.

Problem specification

Given N, L, and Alice's first answer, compute the smallest number of questions Q such that there is a strategy for Peter such that he will surely be able to guess Alice's number in at most Q additional questions.

Gameplay example

Alice: In this game $N = 5$. Guess my number!
Peter: Ok, whatever. **Is your number less than 1?**
Alice: I warn you that in this game I can lie once. **Yes**, it is.
Peter: Either the answer is 0, or she already lied to me. Better to make sure. **Is your number 0 or 2?**
Alice: **No**, it is neither 0, nor 2.
Peter: Oh, so she definitely lied to me already. Too bad I don't know which one of her answers is false. But wait! I'm already sure the answer is not 2. And she can not lie any more. This is starting to be easy. **Is it odd?**
Alice: **No**.
Peter: So it can only be 0 or 4. **Is it 0?**
Alice: **Yes**.
Peter: **Then your number has to be 0!**

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 consists of a single line containing two integers N and L, and a string S. N gives the range of numbers the children play with, L is the threshold from Peter's first question, and S (either "yes" or "no") is Alice's first answer.

Output specification

For each test case, the output shall contain one line with a single integer – the smallest possible number of additional questions Peter needs in order to guarantee that he will be able to guess Alice's number.

Examples

input

```
4

1 47 yes

2 2 yes

2 1 no

5 1 yes
```

output

```
0
3
2
3
```

In the first test case, Peter is sure Alice's number is zero, and thus he needs no more questions.

In the second test case, Alice's answer tells us nothing. Her number is either 0 or 1. In this case, all reasonable questions are equivalent to asking "Is your number 0?". In the worst case, Peter needs to ask this question three times to be sure.

In the third test case, Peter's first question already helped him, and he only needs two more questions to be sure.

The last example is the case from the gameplay example. Peter was actually using one possible optimal strategy.

K equal digits

Every once in a while, Mishka Jabereen sends an interesting mathematical puzzle to his friends. This week's puzzle will be about so-called "repetitive" numbers – the ones whose decimal expansion has just one kind of digit in it. (Examples of such numbers include 7, 11, and 5555.) The puzzle is about finding the largest repetitive number subject to two additional restrictions:

- It must be divisible by at least one number from a given set.
- It can not have more than a given number of digits.

A concrete example of such a problem statement would be: *Find the largest repetitive number with at most 47 digits, which is divisible by 42 or 47!* Mishka is currently playing around with a few such problem statements and he'd like to know all the answers, so that he can choose the nicest one.

Problem specification

A puzzle is described by a number K, the maximal number of digits allowed in the repetitive number, and a set of numbers d_1, d_2, ..., d_R. Your task is to find the greatest repetitive number X that has at most K digits when written in decimal notation, and it is divisible by at least one of the d_i.

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 the numbers K and R. The next R lines contain the numbers d_i.

Output specification

We can describe a repetitive number by specifying its number of digits N and the digit D it contains.

For each test case, the output shall contain one line containing two integers N and D that describe the largest repetitive number that satisfies the conditions from the problem statement.

Example

input

```
3

47 2
42
47

99 4
123
234
345
456

3 1
4700
```

output

```
46 9
96 6
1 0
```

Note that in the third test case "3 0" would not be a correct answer, as "000" is not a valid integer.

Large party

Irena and Sirup are organizing their engagement party next weekend. They want to invite almost everybody. They have just bought a very big round table for this occasion. But they are now wondering how should they distribute people around the table. Irena claimed that when there are more than K women next to each other, this group will chat together for the whole night and won't talk to anybody else.

Sirup had no other choice but to agree with her. However, being a mathematician, he quickly became fascinated by all the possible patterns of men and women around the table.

Problem specification

There will be N people sitting at the round table. Some of them will be men and the rest will be women.

Your task is to count in how many ways it is possible to assign the places to men and women in such a way that there will not be more than K women sitting next to each other.

If one assignment can be made from another one by rotating all the people around the table, we consider them equal (and thus count this assignment only once).

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.

The input for each test case consists of a single line that contains the two integers N and K.

Output specification

For each test case output a single line with one integer – the number of ways how to distribute people around the table, modulo 100 000 007.

Example

input

```
3

3 1

3 3

4 1
```

output

```
2
4
3
```

In the first test case there are two possibilities: MMM or MMW (M is a man, W is a woman).

In the second test case there are two more possibilities: MWW and WWW.

In the third test case the three possibilities are: MMMM, MMMW, and MWMW.