Problem S: Subsequence

Unfortunately, this task has no story because my dog ate some of my notes. Instead, you'll get to help me determine whether I can salvage something out of the scraps left after the dog ran away.

Problem specification

You are given two strings A and B. Find whether B is a subsequence of A.

The subsequence does not need to be contiguous. In other words, your task is to find whether it is possible to change A into B by erasing some (possibly none) of its characters.

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 two lines. The first line contains the string A. The second line contains the string B. Both strings consist only of uppercase English letters.

In the **easy subproblem S1** the string A has length at most 100 and the string B has length 3.

In the **hard subproblem S2** the string A has length at most $100,000$ and the string B has length at most $10,000$. Note that you cannot download the input for subproblem S2 directly. Instead, we have provided a small Python 2 program that will generate the file `s2.in` when executed.

Output specification

For each test case, output a single line with a single word: "YES" or "NO" (quotes for clarity).

Note

In the real contest, some large input files may be provided in the same way as the input `s2.in` in this practice problem. Please make sure you are able to generate it.

Example

input

```
3

DAOBCCCCCGS
DOG

ABCDEF
ACEG

CAT
CTA
```

output

```
YES
NO
NO
```


Problem T: Turning gears

Jano the mechanic has a lot of gears (cogwheels) of different sizes. One day, he took n of them and assembled a complicated machine. Now he wants to know whether and how fast will gear number n rotate if he attempts to rotate gear number 1.

Problem specification

There are n gears numbered from 1 to n. All gears are located in the same plane.

Gear i is represented by a circle with center at (x_i, y_i) and radius r_i. All coordinates are integers. The gears are placed so that the center of each circle lies strictly outside all other circles.

Whenever two circles intersect, the corresponding gears are linked to each other correctly. (A single point of contact between two circles **does count** as an intersection. The size of the intersection does not matter.)

We can describe the rotation of a gear by giving its speed (in full revolutions per second) and its direction of rotation (either clockwise or counterclockwise). Suppose we have two linked gears: one with radius r_1 and the other with radius r_2. If the first gear rotates at v_1 revolutions per second, the other gear will also rotate, but in the opposite direction and possibly at a different speed v_2. The speed v_2 can be computed as follows: $v_2 = v_1 r_1 / r_2$.

In a machine with multiple gears the above equation applies to each pair of linked gears. If this would mean a gear has to rotate at two different speeds or directions, it won't rotate at all.

Jano will attempt to rotate gear number 1 clockwise at 1 revolution per second. How will gear n rotate?

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 number n ($2 \leq n \leq 1\,000$) – the number of gears in the machine. The i-th of the next n lines describes gear i. The description consists of three **integers** x_i, y_i and r_i – its center and its radius, as described above. You may assume that $0 \leq x_i, y_i \leq 50\,000$ and $1 \leq r_i \leq 1\,000$.

In the **subproblem T1** you can make the following additional assumptions about each test case:

- The number of gears will not exceed 200.
- All gears will be directly or indirectly linked to gear 1.
- There will be exactly $n - 1$ pairs of directly linked gears.

Output specification

For each test case, output a single line containing the speed and the direction of rotation of gear n. The speed should be expressed as a fraction `p/q` in reduced form. (I.e., the greatest common divisor of p and q should be 1.) The direction should be either "`clockwise`" or "`counterclockwise`".

If the n-th gear will not rotate at all, print "`does not rotate`" instead.

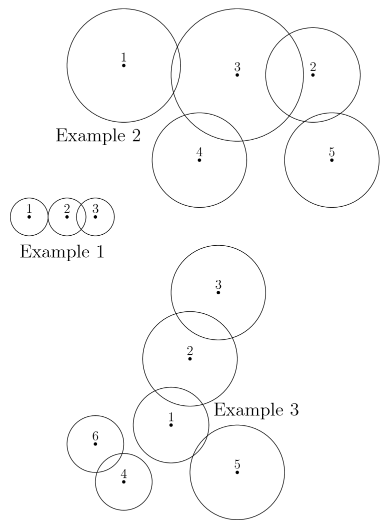
Example

input

```
3

3
0 0 2
4 0 2
7 0 2

5
0 10 6
20 9 5
12 9 7
8 0 5
22 0 5

6
8 6 4
10 13 5
13 20 5
3 0 3
15 1 5
0 4 3
```

output

```
1/1 clockwise
6/5 counterclockwise
does not rotate
```



In the first test case note that gears 1 and 2 are linked, even though they only touch each other at a single point. Also note that the fraction is printed as "1/1": the denominator is always printed, even if it is 1. The outputs "2/2" and "1" would both be rejected by the grader.

In the second test case we can compute the answer as follows:

- *Gear 1 rotates clockwise at 1 revolution per second (rps).*
- *Gear 1 is directly linked to gear 3, thus gear 3 rotates counterclockwise at 6/7 rps.*
- *Gear 3 is also linked to gears 2 and 4. Thus, both of these gears will rotate clockwise. As both of them have the same radius (5), they will both rotate at the same speed: 6/5 rps.*
- *Gear 5 is linked to gear 2. Therefore, gear 5 will rotate counterclockwise. As gears 2 and 5 have the same radius, gear 5 will also rotate at 6/5 rps.*

The third test case ***does not satisfy*** *the additional constraints for subproblem T1. Hence, this test case can only appear in the subproblem T2. This test case illustrates one of the two main reasons why the answer can be "does not rotate".*

Problem U: Ultimate magic rectangles

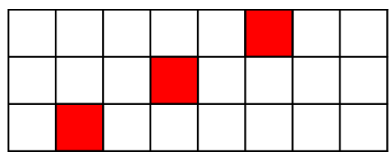
Bob is busy today, so Alice has found a single-player game.

Problem specification

In this game, Alice is given an integer s and an empty table with $r = 3$ rows and c columns. Alice has to fill in some **nonnegative integers** into the cells of the table.

Three cells are called a *triplet* if they lie in different rows and their centers lie on a straight line. The goal of the game is to fill the table in such a way that each triplet will have the same sum.

You are given the number of columns c and the desired sum of each triplet s. Compute the number of ways to fill the table in the desired way. Since this number may be large, compute it modulo $10^9 + 9$.



Above: one of the many triplets on a board with $c = 8$ columns.

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 one line containing two space-separated integers c and s.

In the **easy subproblem U1**, $1 \leq c \leq 50$ and $0 \leq s \leq 50$.

In the **hard subproblem U2**, $1 \leq c \leq 1000$ and $0 \leq s \leq 10^9$.

Output specification

For each test case, print one integer on a separate line – the number of solutions, modulo $10^9 + 9$.

Example

input

```
2

3 1

4 3
```

output

```
5
34
```

In the first test case there are five triplets: each column and both main diagonals of the 3×3 square. The sum of each triplet must be 1, which means that each triplet must contain two 0s and a 1. These are the five solutions:

```
111   000   000   101   010
000   111   000   000   000
000   000   111   010   101
```

In the second test case one of the 34 valid solutions is a 3×4 rectangle full of 1s.

