Problem A
Zipper

Input: A.txt

Summer Camp (1st Day), Tokyo
23 Sep 2005

Given three strings, you are to determine whether the third string can be formed by combining the characters in the first two strings. The first two strings can be mixed arbitrarily, but each must stay in its original order.

For example, it is possible to form “tcraete” from “cat” and “tree” by alternating characters from the two strings. It is also possible to form “catrtee” from “cat” and “tree”. However, it is impossible to form “cttaree” from “cat” and “tree”.

Input

The first line contains a single positive integer from 1 through 1000. It represents the number of data sets to follow. The processing for each data set is identical. The data sets appear on the following lines, one data set per line. For each data set, the line of input consists of three strings, separated by a single space. All strings are composed of upper and lower case letters only. The length of the third string is always the sum of the lengths of the first two strings. The first two strings will have lengths between 1 and 200 characters, inclusive.

Output

For each data set, print:

```
Data set n: yes
```

if the third string can be formed from the first two, or

```
Data set n: no
```

if it cannot. Of course `n` should be replaced by the data set number. See the sample output below for an example.

Sample Input

```
3
cat tree tcraete
cat tree catrtee
cat tree cttaree
```

Output for the Sample Input

```
Data set 1: yes
Data set 2: yes
Data set 3: no
```

Problem B
Ellipse Intersection

Summer Camp (1st Day), Tokyo
23 Sep 2005

Input: B.txt

Recently the astronomers have discovered a peculiar pair of planets, named A and B. As we know, a planet usually moves in an ellipse orbit, so do A and B. But their orbits are quite special:

1. Their orbits are in the same plane and share the same center;
2. The segments formed by joining their own focuses respectively are perpendicular to each other.

If we denote the center as point O, and the focuses of orbit A as F_1 and F_2, then we can build Cartesian coordinates, with point O being the origin, and the line through F_1 and F_2 being the x-axis.

Here is a sample:

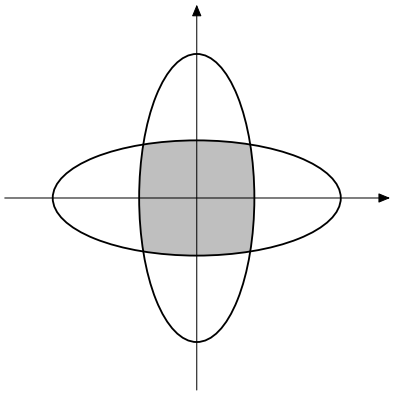

Figure 1: Sample orbits

As the astronomers would like to know more about the planets, they decide to calculate the intersection area first. But unfortunately calculating the intersection area is so complex and beyond their ability, so they have to turn to you, a talent of programming.

You are given the description of two ellipse orbits satisfying the conditions above, and required to calculate their intersection area.

Input

Input may contain multiple test cases. The first line is a positive integer n ($n \leq 100$), denoting the number of test cases below. Each case is composed of two lines, the first one is the description of orbit A, and the second one the description of orbit B. Each description contains two positive integers, a and b ($a, b \leq 100$), denoting the ellipse equation $x^2/a^2 + y^2/b^2 = 1$. It is guaranteed that the focuses of orbit A are on x-axis, and the focuses of orbit B are on y-axis.

Output

For each test case, output one line containing a single real number, the area of the intersection, with an error up to 0.001.

Sample Input

```
1
2 1
1 2
```

Output for the Sample Input

```
3.709
```

Problem C
Garbage Collection

Input: C.txt

Summer Camp (1st Day), Tokyo
23 Sep 2005

Modern programming languages include automatic memory management systems, known as “garbage collection” or “GC”, to simplify the task of programmers.

Many diverse algorithms of GC has been proposed. At the very least, all of them have a common principle: “Do not free memory that is possible to be referred later in the running program. Just free memory that will never be used.” This means, from another point of view, that the core of GC is an algorithm to classify whether a memory block may still be referred in a program (*alive* memory) or not (*dead* memory). However, it is not possible, in general, for a garbage collector to determine exactly which blocks are still alive. All garbage collectors hence use some efficient approximation to liveness.

Your friend Prof. George Collins is a researcher of garbage collection. One day, he came up with an new idea for efficient approximation to the liveness classification. Here is his approximation:

- A memory block directly pointed from active local variables, which are the variables declared in functions not finished yet, are all considered to be *alive*.
- If once a pointer in memory block *N* points another memory block *M*, life durations of these two blocks are equated. In other words, *N* is alive if and only if *M* is alive. This may seem odd, as there can be a situation in which *M* is referrable from the program but *N* is not. It is completely true, but for an *approximation* of liveness, over classification to *alive* is not a fatal problem. Prof. Collins believes that this approximation gives a significant speed up of garbage collectors.
- Any other memory blocks that cannot be determined to be *alive* in the two rules above are *dead*.

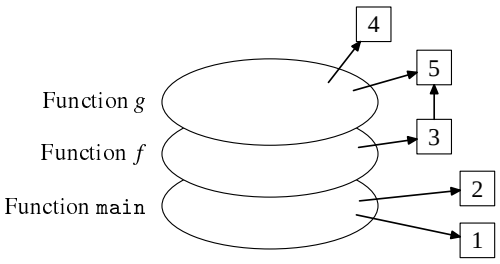

Figure 2: An example state with 3 stack frames and 5 objects.

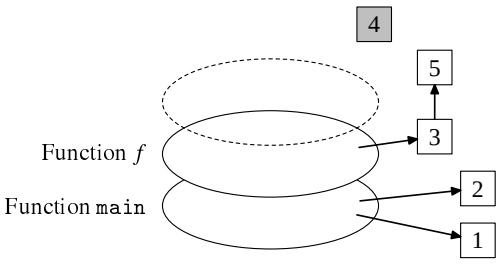

Figure 3: The object 4 went *dead* after returning from function *g*.

Since Prof. Collins is not good at programming, he asked you to write a program to test the performance of his new algorithm. He needs a program which reads a sequence of machine instructions to be executed and logs the number of memory blocks classified as *dead* by his new algorithm.

The input is a sequence of instructions. For example:

```
alloc
call
alloc
alloc
link 1 2
return
return
```

There are four kinds of instructions:

Instruction	Description
`alloc`	Allocates a new memory block and point it from a new local variable in the current function.
`link` *N M*	Points the *M*-th allocated block from the *N*-th allocated block ($N, M > 0$).
`call`	Calls a function.
`return`	Returns from the current function.

Your program should output the number of memory blocks newly classified as *dead* by Prof. Collins' method on each `return` instruction.

```
1
2
```

In case of the sample input above, on the first `return` the 3rd memory block goes `dead` because its depending function is terminated by the `return` instruction. The 2nd block stays *alive* since it is pointed by the 1st block, which is still in an active stack frame. Therefore, 1 must be printed. The next `return` kills the 1st and the 2nd blocks, thus the output is 2.

Input

Multiple test cases are contained in the input of this problem. Each case begins with a line containing a single integer L ($1 \leq L \leq 100000$), which is the length of the instruction sequence. Then L lines follow. Each line contains one instruction in the form as stated above.

All instruction sequences in the input are valid. That is:

- Both arguments of a `link` instruction are *alive* memory blocks.
- All but last `return` instruction have a corresponding `call` instruction before them.
- The last instruction is always `return`, which indicates the end of program.

The input ends with a line containing just a single 0.

Output

For every test case, first output a line with the number of the test case as shown in the sample. Then print a line containg the number of new *dead* blocks on each `return` instruction in the input sequence.

Sample Input

```
7
alloc
call
alloc
alloc
link   1 2
return
return
6
alloc
call
alloc
link   2 1
return
return
0
```

Output for the Sample Input

```
Program #1
1
2
Program #2
0
2
```

Problem D
Painting

Input: D.txt

Summer Camp (1st Day), Tokyo
23 Sep 2005

You have figures that consist of white circles (○) and black circles (●) locating in the shape of the regular triangle. An example of such figures is as follows.

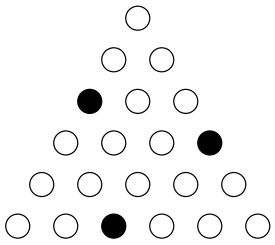

Figure 4: An example triangle

Now let us consider painting all the white circles into red. You can paint white circles in one step, if the white circles are on a line in parallel to one of the edges of the triangle. In order words, you can paint either from left to right, from upper-right to lower-left, or upper-right to lower-left.

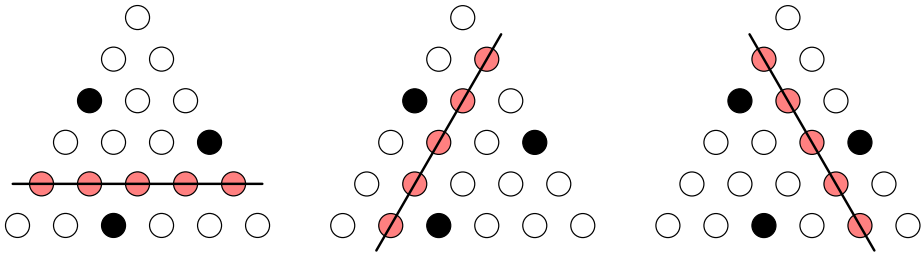

Figure 5: Three kinds of steps

You can paint white circles more than once, but you must not paint any black circles (i.e. the line must not meet the black circles).

The problem is that how many steps do you need to paint all the white circles in the given figures. In the case of the example above we need six steps to paint all the white circles, where one of such painting is as follows.

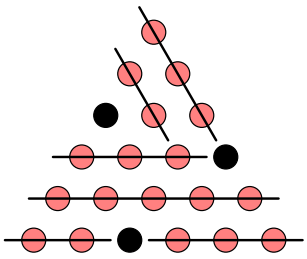

Figure 6: An example way to paint all circles in 6 steps

Input

Input consists of multiple test cases.

The first line of each case contains a single positive integer N ($N \leq 15$) that specifies the size of triangles. The following N lines represent a figure; the i-th line contains i characters of either `w` or `b`, where `w` denotes a white circle and `b` denotes a black circle.

Input is terminated by a line that contains a single zero, which should not be processed.

Output

For each case, your program should print the minimum number of steps to paint all the white circles in one line.

Sample Input

```
6
w
ww
bww
wwwb
wwwww
wwbwww
0
```

Output for the Sample Input

```
6
```

Problem E
Building Bridges

Input: E.txt

Summer Camp (1st Day), Tokyo
23 Sep 2005

Irene B. Moore is working as a mayor of the City Palau, which has a number of islands. Some bridges are built between islands, but not all islands are connected by those bridges. People therefore move between islands by ship if needed. However, because the number of seamen is decreasing in these days, people living there began to demand her to build more bridges so they can move among all islands without ship. Irene is willing to meet their demands.

Unfortunately, the city is not in so easy circumstances. Irene thus decided to build the least number of new bridges required to make people possible to move between every pair of islands by land. Moreover, she decided to shorten the total distance of new bridges, intending to minimize the construction cost.

In order to carry out her idea, Irene told Christopher, a person working under her, to make an estimate of the shortest total distance of bridges to be built. For estimation, he modeled the island area as follows. First, he considered the area to be a two-dimensional plane. Secondly, he regarded islands as circular. Lastly, he made an assumption that a bridge is a line segment connecting two islands in the shortest distance.

He knew that it is possible to compute the shortest total distance of new bridges based on his model, but he did not know how to do that. So he called you for help.

Your task is to write a program that computes the distance, given the data of the islands and the existing bridges. Note that a bridge crossing over an island or another bridge may *not* be built.

Input

Input consists of multiple data sets.

The first line of each data set contains a single positive integer n ($2 \leq n \leq 50$) that represents the number of islands in the City Palau. The following n lines describe the islands. The i-th line contains three real numbers x_i ($-100 \leq x_i \leq 100$), y_i ($-100 \leq y_i \leq 100$) and r_i ($1 \leq r_i \leq 10$), where (x_i, y_i) and r_i denotes the center and radius of the i-th island respectively. After that, there is a line containing a single positive integer m that represents the number of existing bridges. The following m lines describe the bridges. Each line contains two integers s_j and t_j, indicating a bridge has already been built between the s_j-th and the t_j-th islands.

You may assume that no two islands are close within the distance of one. It is also guaranteed that no existing bridge crosses over an island or another existing bridge.

Input is terminated by a line containing a single zero. This line is not a part of data set.

Output

For each data set, print the shortest total distance in a line. Each number may be printed with an arbitrary number of decimal digits, but may not contain an error greater than 0.001.

There may be test cases that no new bridges are required to be built. In such cases, just print zero as the distance.

No extra space or blank line should appear.

Sample Input

```
4
5.0 5.0 1.0
0.0 5.0 1.0
0.0 0.0 1.0
5.0 0.0 1.0
2
1 2
3 4
0
```

Output for the Sample Input

```
3.000
```