The 26th Annual

ACM International Collegiate Programming Contest ASIA Regional - Taejon

Problem A

Calendar Game

Input: calen.in

Adam and Eve enter this year's ACM International Collegiate Programming Contest. Last night, they played the Calendar Game, in celebration of this contest. This game consists of the dates from January 1, 1900 to November 4, 2001, the contest day. The game starts by randomly choosing a date from this interval. Then, the players, Adam and Eve, make moves in their turn with Adam moving first: Adam, Eve, Adam, Eve, etc. There is only one rule for moves and it is simple: from a current date, a player in his/her turn can move either to the next calendar date or the same day of the next month. When the next month does not have the same day, the player moves only to the next calendar date. For example, from December 19, 1924, you can move either to December 20, 1924, the next calendar date, or January 19, 1925, the same day of the next month. From January 31 2001, however, you can move only to February 1, 2001, because February 31, 2001 is invalid.

A player wins the game when he/she exactly reaches the date of November 4, 2001. If a player moves to a date after November 4, 2001, he/she looses the game.

Write a program that decides whether, given an initial date, Adam, the first mover, has a winning strategy.

For this game, you need to identify leap years, where February has 29 days. In the Gregorian calendar, leap years occur in years exactly divisible by four. So, 1993, 1994, and 1995 are not leap years, while 1992 and 1996 are leap years. Additionally, the years ending with 00 are leap years only if they are divisible by 400. So, 1700, 1800, 1900, 2100, and 2200 are not leap years, while 1600, 2000, and 2400 are leap years.

Input

The input consists of T test cases. The number of test cases (T) is given in the first line of the input file. Each test case is written in a line and corresponds to an initial date. The three integers in a line, `YYYY MM DD`, represent the date of the `DD`-th day of `MM`-th month in the year of `YYYY`. Remember that initial dates are randomly chosen from the interval between January 1, 1900 and November 4, 2001.

Output

Print exactly one line for each test case. The line should contain the answer "YES" or "NO" to the question of whether Adam has a winning strategy against Eve. Since we have T test cases, your program should output totally T lines of "YES" or "NO".

Sample Input (calen.in)	**Output for the Sample Input**
3 2001 11 3 2001 11 2 2001 10 3	YES NO NO

The 26th Annual

ACM International Collegiate Programming Contest ASIA Regional - Taejon

Problem B

Wooden Sticks

Input: stick.in

There is a pile of n wooden sticks. The length and weight of each stick are known in advance. The sticks are to be processed by a woodworking machine in one by one fashion. It needs some time, called setup time, for the machine to prepare processing a stick. The setup times are associated with cleaning operations and changing tools and shapes in the machine. The setup times of the woodworking machine are given as follows:

(a) The setup time for the first wooden stick is 1 minute.
(b) Right after processing a stick of length l and weight w, the machine will need no setup time for a stick of length l' and weight w' if $l \le l'$ and $w \le w'$. Otherwise, it will need 1 minute for setup.

You are to find the minimum setup time to process a given pile of n wooden sticks. For example, if you have five sticks whose pairs of length and weight are $(4,9)$, $(5,2)$, $(2,1)$, $(3,5)$, and $(1,4)$, then the minimum setup time should be 2 minutes since there is a sequence of pairs $(1,4)$, $(3,5)$, $(4,9)$, $(2,1)$, $(5,2)$.

Input

The input consists of T test cases. The number of test cases (T) is given in the first line of the input file. Each test case consists of two lines: The first line has an integer n, $1 \le n \le 5000$, that represents the number of wooden sticks in the test case, and the second line contains $2n$ positive integers l_1, w_1, l_2, w_2, …, l_n, w_n, each of magnitude at most 10000, where l_i and w_i are the length and weight of the ith wooden stick, respectively. The $2n$ integers are delimited by one or more spaces.

Output

The output should contain the minimum setup time in minutes, one per line.

Sample Input (stick.in)	Output for the Sample Input
3 5 4 9 5 2 2 1 3 5 1 4 3 2 2 1 1 2 2 3 1 3 2 2 3 1	2 1 3

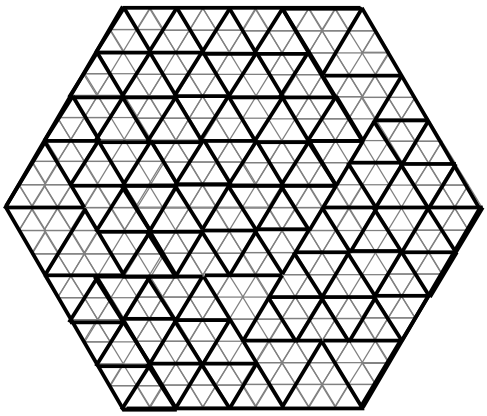
Problem C

~~Problem B~~

(Program filename: B.CPP or B.PAS or B.DPR or B.java)

The Bermuda Triangle

People in the hidden region of the Bermuda Triangle make everything they need in triangular shapes. One day, someone decided to break the rule and bake a hexagonally shaped cake. But as usual, he has to serve the cake in triangular pieces. The pieces are equilateral triangles but in different sizes for different people. He can use as many triangles as needed to cut the cake into pieces, such that nothing remains from the cake. For example, the following figure shows one way that a hexagon with side 9 can be cut into triangles with side 2 and 3. (The cake is cut along the thick lines, thin lines are drawn to show the sizes).



Input is a hexagon and triangle types (specified by the length of their sides) and the goal is to decide if the hexagon can be completely divided by the given triangle types.

Input (filename: B.IN)

The first line of the input file contains a single integer t ($1 \le t \le 10$), the number of test cases, followed by the input data for each test case. Each test case consists of a single line, containing s ($1 \le s \le 25$), the length of the hexagon's side, followed by n, the number of triangle types ($1 \le n \le 10$), followed by n integers representing the length of each triangle type's side (between 1 and 25, inclusive).

Output (filename: B.OUT)

There should be one output line per test case containing either `YES` or `NO` depending on whether the hexagon can be completely divided by the given triangle types.

Sample Input

```
3
5 2 2 3
7 2 3 2
13 2 2 3
```

Sample Output

```
NO
NO
YES
```

ACM International Collegiate Programming Contest
Japan Domestic, 2005-07-01

Problem D
Traveling by Stagecoach

Once upon a time, there was a traveler.

He plans to travel using stagecoaches (horse wagons). His starting point and destination are fixed, but he cannot determine his route. Your job in this problem is to write a program which determines the route for him.

There are several cities in the country, and a road network connecting them. If there is a road between two cities, one can travel by a stagecoach from one of them to the other. A coach ticket is needed for a coach ride. The number of horses is specified in each of the tickets. Of course, with more horses, the coach runs faster.

At the starting point, the traveler has a number of coach tickets. By considering these tickets and the information on the road network, you should find the best possible route that takes him to the destination in the shortest time. The usage of coach tickets should be taken into account.

The following conditions are assumed.

- A coach ride takes the traveler from one city to another directly connected by a road. In other words, on each arrival to a city, he must change the coach.
- Only one ticket can be used for a coach ride between two cities directly connected by a road.
- Each ticket can be used only once.
- The time needed for a coach ride is the distance between two cities divided by the number of horses.
- The time needed for the coach change should be ignored.

Input

The input consists of multiple datasets, each in the following format. The last dataset is followed by a line containing five zeros (separated by a space).

$n\ m\ p\ a\ b$
$t_1\ t_2\ \ldots\ t_n$
$x_1\ y_1\ z_1$
$x_2\ y_2\ z_2$
...
$x_p\ y_p\ z_p$

Every input item in a dataset is a non-negative integer. If a line contains two or more input items, they are separated by a space.

n is the number of coach tickets. You can assume that the number of tickets is between 1 and 8. m is the number of cities in the network. You can assume that the number of cities is between 2 and 30. p is the number of roads between cities, which may be zero.

a is the city index of the starting city. *b* is the city index of the destination city. *a* is not equal to *b*. You can assume that all city indices in a dataset (including the above two) are between 1 and *m*.

The second line of a dataset gives the details of coach tickets. t_i is the number of horses specified in the *i*-th coach ticket (1<=*i*<=*n*). You can assume that the number of horses is between 1 and 10.

The following *p* lines give the details of roads between cities. The *i*-th road connects two cities with city indices x_i and y_i, and has a distance z_i (1<=*i*<=*p*). You can assume that the distance is between 1 and 100.

No two roads connect the same pair of cities. A road never connects a city with itself. Each road can be traveled in both directions.

Output

For each dataset in the input, one line should be output as specified below. An output line should not contain extra characters such as spaces.

If the traveler can reach the destination, the time needed for the best route (a route with the shortest time) should be printed. The answer should not have an error greater than 0.001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

If the traveler cannot reach the destination, the string "`Impossible`" should be printed. One cannot reach the destination either when there are no routes leading to the destination, or when the number of tickets is not sufficient. Note that the first letter of "`Impossible`" is in uppercase, while the other letters are in lowercase.

Sample Input

```
3 4 3 1 4
3 1 2
1 2 10
2 3 30
3 4 20
2 4 4 2 1
3 1
2 3 3
1 3 3
4 1 2
4 2 5
2 4 3 4 1
5 5
1 2 10
2 3 10
3 4 10
1 2 0 1 2
1
8 5 10 1 5
2 7 1 8 4 5 6 3
1 2 5
2 3 4
3 4 7
4 5 3
1 3 25
2 4 23
3 5 22
1 4 45
```

```
2 5 51
1 5 99
0 0 0 0 0
```

Output for the Sample Input

```
30.000
3.667
Impossible
Impossible
2.856
```

Since the number of digits after the decimal point is not specified, the above result is not the only solution. For example, the following result is also acceptable.

```
30.0
3.66667
Impossible
Impossible
2.85595
```

First Input Data

Here is your first input.

The ACM ICPC

ACM International Collegiate Programming Contest
Japan Domestic, 2005-07-01

Problem E
Earth Observation with a Mobile Robot Team

A new type of mobile robot has been developed for environmental earth observation. It moves around on the ground, acquiring and recording various sorts of observational data using high precision sensors. Robots of this type have short range wireless communication devices and can exchange observational data with ones nearby. They also have large capacity memory units, on which they record data observed by themselves and those received from others.

Figure 1 illustrates the current positions of three robots A, B, and C and the geographic coverage of their wireless devices. Each circle represents the wireless coverage of a robot, with its center representing the position of the robot. In this figure, two robots A and B are in the positions where A can transmit data to B, and vice versa. In contrast, C cannot communicate with A or B, since it is too remote from them. Still, however, once B moves towards C as in Figure 2, B and C can start communicating with each other. In this manner, B can relay observational data from A to C. Figure 3 shows another example, in which data propagate among several robots instantaneously.

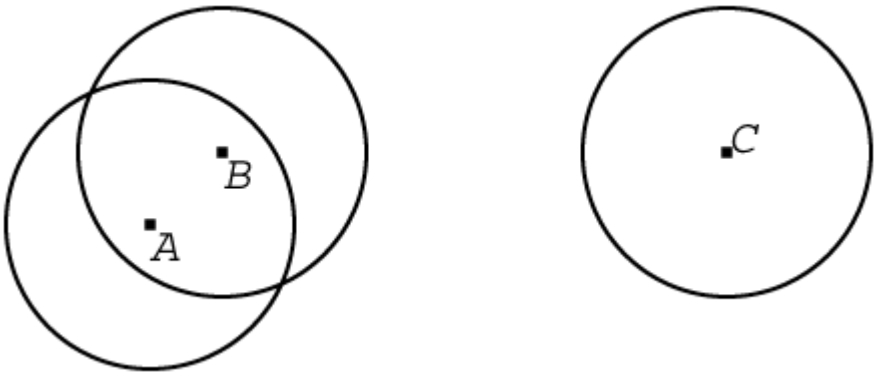


Figure 1: The initial configuration of three robots

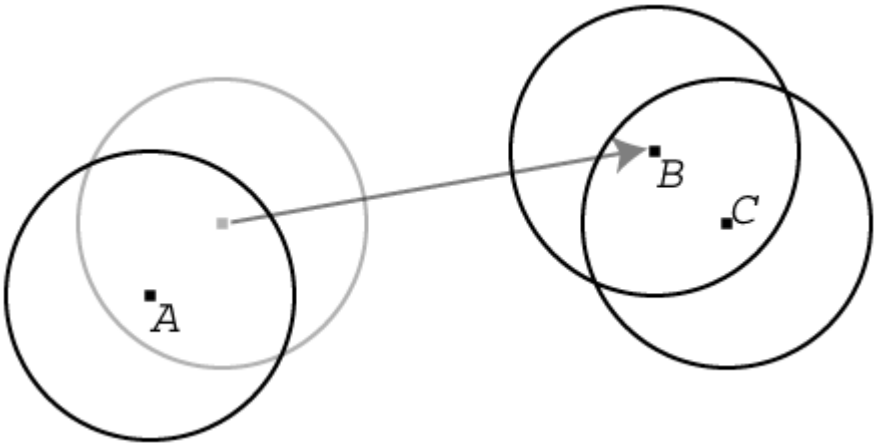


Figure 2: Mobile relaying

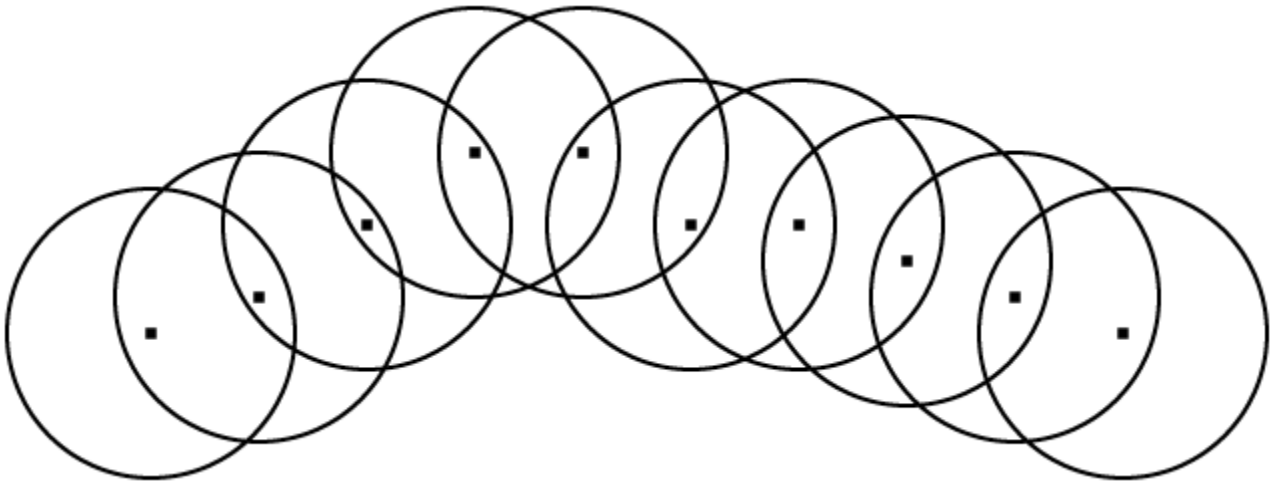


Figure 3: Instantaneous relaying among multiple robots

As you may notice from these examples, if a team of robots move properly, observational data quickly spread over a large number of them. Your mission is to write a program that simulates how information spreads among robots. Suppose that, regardless of data size, the time necessary for communication is negligible.

Input

The input consists of multiple datasets, each in the following format.

> *N T R*
> *nickname and travel route of the first robot*
> *nickname and travel route of the second robot*
> ...
> *nickname and travel route of the N-th robot*

The first line contains three integers *N*, *T*, and *R* that are the number of robots, the length of the simulation period, and the maximum distance wireless signals can reach, respectively, and satisfy that 1 <=*N* <= 100, 1 <= *T* <= 1000, and 1 <= *R* <= 10.

The *nickname and travel route* of each robot are given in the following format.

> *nickname*
> $t_0\ x_0\ y_0$
> $t_1\ vx_1\ vy_1$
> $t_2\ vx_2\ vy_2$
> ...
> $t_k\ vx_k\ vy_k$

Nickname is a character string of length between one and eight that only contains lowercase letters. No two robots in a dataset may have the same nickname. Each of the lines following *nickname* contains three integers, satisfying the following conditions.

> $0 = t_0 < t_1 < ... < t_k = T$
> $-10 <= vx_1, vy_1, ..., vx_k, vy_k <= 10$

A robot moves around on a two dimensional plane. (x_0, y_0) is the location of the robot at time 0. From time t_{i-1} to t_i ($0 < i <= k$), the velocities in the x and y directions are vx_i and vy_i, respectively. Therefore, the travel route of a robot is piecewise linear. Note that it may self-overlap or self-intersect.

You may assume that each dataset satisfies the following conditions.

- The distance between any two robots at time 0 is not exactly R.
- The x- and y-coordinates of each robot are always between -500 and 500, inclusive.
- Once any robot approaches within $R + 10^{-6}$ of any other, the distance between them will become smaller than $R - 10^{-6}$ while maintaining the velocities.
- Once any robot moves away up to $R - 10^{-6}$ of any other, the distance between them will become larger than $R + 10^{-6}$ while maintaining the velocities.
- If any pair of robots mutually enter the wireless area of the opposite ones at time t and any pair, which may share one or two members with the aforementioned pair, mutually leave the wireless area of the opposite ones at time t', the difference between t and t' is no smaller than 10^{-6} time unit, that is, $|t - t'| >= 10^{-6}$.

A dataset may include two or more robots that share the same location at the same time. However, you should still consider that they can move with the designated velocities.

The end of the input is indicated by a line containing three zeros.

Output

For each dataset in the input, your program should print the nickname of each robot that have got until time T the observational data originally acquired by the first robot at time 0. Each nickname should be written in a separate line in dictionary order without any superfluous characters such as leading or trailing spaces.

Sample Input

```
3 5 10
red
0 0 0
5 0 0
green
0 5 5
5 6 1
blue
0 40 5
5 0 0
3 10 5
atom
0 47 32
5 -10 -7
10 1 0
pluto
0 0 0
7 0 0
10 3 3
gesicht
0 25 7
5 -7 -2
10 -1 10
4 100 7
impulse
```

```
0 -500 0
100 10 1
freedom
0 -491 0
100 9 2
destiny
0 -472 0
100 7 4
strike
0 -482 0
100 8 3
0 0 0
```

Output for the Sample Input

```
blue
green
red
atom
gesicht
pluto
freedom
impulse
strike
```

First Input Data

Here is your first input.

The ACM ICPC

Southeastern European Regional Programming Contest
Bucharest, Romania
October 21, 2000

Problem F
Rectangles

Input File: F.DAT
Program Source File: F.PAS or F.C or F.CPP

A specialist in VLSI design testing must decide if there are some components that cover each other for a given design. A component is represented as a rectangle. Assume that each rectangle is rectilinearly oriented (sides parallel to the x and y axis), so that the representation of a rectangle consists of its minimum and maximum x and y coordinates.

Write a program that counts the rectangles that are entirely covered by another rectangle.

The input file contains the text description of several sets of rectangles. The specification of a set consists of the number of rectangles in the set and the list of rectangles given by the minimum and maximum x and y coordinates separated by white spaces, in the format:

nr_rectangles
$xmin_1$ $xmax_1$ $ymin_1$ $ymax_1$
$xmin_2$ $xmax_2$ $ymin_2$ $ymax_2$
….
$xmin_n$ $xmax_n$ $ymin_n$ $ymax_n$

The output should be printed on the standard output. For each given input data set, print one integer number in a single line that gives the result (the number of rectangles that are covered). An example is given in Figure 1.

input	output
3 100 101 100 101 0 3 0 101 20 40 10 400 4 10 20 10 20 10 20 10 20 10 20 10 20 10 20 10 20	0 4

Figure 1

The 26th Annual

ACM International Collegiate Programming Contest ASIA Regional - Taejon

Problem G
Farmland
Input: farmland.in

We have a map for farming land in a country. The whole farming land of the country is divided into a set of disjoint farming regions. Each farmer owns only one farming region in this country. There is a boundary fence between two neighboring farming regions. The farmland map for this country can be represented in a plane graph. The following Figure-1 shows one example.

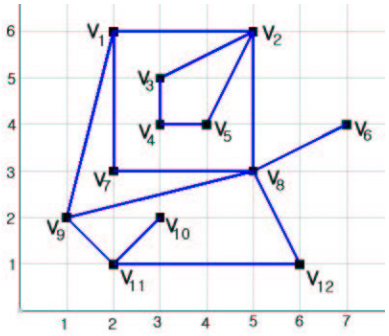


Figure-1: Farmland graph G(V,E)

There are two types of edges, boundary edge and non-boundary edge. All edges of G(V,E) except (v_8, v_6) and (v_{11}, v_{10}) are boundary edges which are between two neighboring farming regions. The "proper farming region" in a Farmland graph is a closed region bounded by a simple cycle and it should not contain any vertices or edges inside. In this figure, the polygon <v_1,v_9,v_8,v_7 > is a proper farming region, and the region < v_2, v_1, v_7, v_8 , v_2, v_5, v_4, v_3 > is not a proper farming region since its boundary cycle is not simple.

We assume that the farmland graph G(V,E) is a simple connected graph, which does not allow self-loops (Figure-2 (a)) and parallel edges (Figure-2 (b)). Also in Farmland graph G(V,E), we do not consider the outer face of G(V,E). You can see that there are 2 proper farming regions in G(V,E) shown in Figure-1, namely <v_1,v_9,v_8,v_7> and <v_2,v_3,v_4,v_5>, since there are no vertices or edges inside. But the polygon<v_1,v_7,v_8,v_2> is not a proper farming region since vertex v_3, v_4, and v_5 are located in that region. Similarly, the region <v_9,v_{11},v_{12},v_8> is not a proper region because a vertex v_{10} is inside the region. A degenerate polygon <v_6, v_8> is not a proper region because it has no valid area inside.

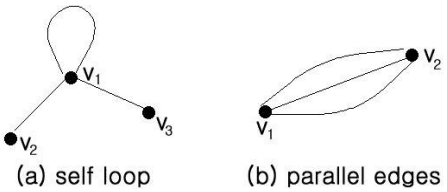


Figure-2: (a) self-loop <v_1,v_1> , and (b) 3 parallel edges { <v_1,v_2>, <v_1,v_2>, <v_1,v_2> }

There are other assumptions for input farmland graph data.

1. There is at least one proper farming region.
2. The position of each vertex in Farmland graph is distinct.
3. There is no edge crossing, which means the graph G(V,E) is a plane graph.
4. Farmland graph G(V,E) is simple and connected.

Let us define the "size" of proper farming region. The size of proper farming region is the number of boundary edges of that region. For example, the size of the proper farming region $<v_2,v_3,v_4,v_5>$ is 4.

The problem is to find the number of proper regions that have a specified size. If you are requested to find the number of proper regions with size of 4 in the graph given in Figure-1, you must answer that there are 2 proper regions whose sizes are 4 because farming regions $< v_1,v_9,v_8,v_7 >$ and $<v_2,v_3,v_4,v_5>$ are proper regions and their sizes are 4. If there are no such regions, then you have to print 0.

Input

The input file consists of *M* test cases. The first line of the input file contains a positive integer *M*, the number of test cases you are to solve. After the first line, input data for *M* cases follow. The first line of each test case contains a positive integer *N* (≥ 3), the number of vertices. Each of the following *N* lines is of the form:

$$i \;\; x_i \;\; y_i \;\; d_i \;\; a_1 \; a_2 \; a_3 \;\; ..... \; a_{di}$$

"*i*" is the vertex number, x_i and y_i are the coordinate (x_i, y_i) of the vertex *i*, and d_i is the degree of the vertex *i*. The following { a_i } are the adjacent vertices of the vertex *i*. The last line gives *k*, the size of proper regions that you have to count.

Note that *M,* the number of cases in input file is less than 10. *N*, the number of vertices of a given farmland graph is less than 200. All vertices are located on grid points of the 1000 x 1000 lattice grid.

Output

The output must contain *M* non-negative integers. Each line contains the answer *n* to the corresponding case of the input file.

Sample Input (farmland.in)	**Output for the Sample Input**
2 12 1 2 6 3 9 7 2 2 5 6 4 5 3 1 8 3 3 5 2 4 2 4 3 4 2 3 5 5 4 4 2 4 2 6 7 4 1 8 7 2 3 2 8 1 8 5 3 5 7 2 9 12 6 9 1 2 3 11 8 1 10 3 2 1 11 11 2 1 3 10 9 12 12 6 1 2 8 11 4 3 1 2 2 2 2 3 2 1 1 2 1 3 3 4 1 2 1 2 4	2 0

Southeastern European Regional Programming Contest
Bucharest, Romania
October 21, 2000

Problem H
~~Problem D~~
Girls and Boys

Input File: D.DAT
Program Source File: D.PAS or D.C or D.CPP

In the second year of the university somebody started a study on the romantic relations between the students. The relation “romantically involved” is defined between one girl and one boy. For the study reasons it is necessary to find out the maximum set satisfying the condition: there are no two students in the set who have been “romantically involved”. The result of the program is the number of students in such a set.

The input file contains several data sets in text format. Each data set represents one set of subjects of the study, with the following description:

the number of students

the description of each student, in the following format

student_identifier:(number_of_romantic_relations) student_identifier1 **student_identifier**2 **student_identifier**3 .....

or

student_identifier:(0)

The student_identifier is an integer number between **0** and **n-1**, for **n** subjects.

For each given data set, the program should write to standard output a line containing the result.

An example is given in Figure 1.

input	output
7 0: (3) 4 5 6 1: (2) 4 6 2: (0) 3: (0) 4: (2) 0 1 5: (1) 0 6: (2) 0 1 3 0: (2) 1 2 1: (1) 0 2: (1) 0	5 2

Figure 1