



Problem A Strategic game

Input File: A.DAT

Program Source File: A.PAS or A.C or A.CPP

Bob enjoys playing computer games, especially strategic games, but sometimes he cannot find the solution fast enough and then he is very sad. Now he has the following problem. He must defend a medieval city, the roads of which form a tree. He has to put the minimum number of soldiers on the nodes so that they can observe all the edges. Can you help him?

Your program should find the minimum number of soldiers that Bob has to put for a given tree.

The input file contains several data sets in text format. Each data set represents a tree with the following description:

- the number of nodes
- the description of each node in the following format

`node_identifier:(number_of_roads) node_identifier1 node_identifier2 ...`

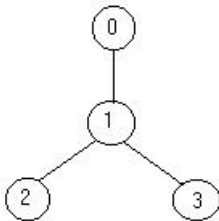
`node_identifier`_{number_of_roads}

or

`node_identifier:(0)`

The node identifiers are integer numbers between 0 and n-1, for n nodes (0 < n ≤ 1500). Every edge appears only once in the input data.

For example for the tree:



the solution is one soldier (at the node 1).

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 minimum number of soldiers). An example is given in the following table:

Input	Output
4	1
0:(1) 1	2
1:(2) 2 3	
2:(0)	
3:(0)	
5	
3:(3) 1 4 2	
1:(1) 0	

2: (0)	
0: (0)	
4: (0)	



Southeastern European Regional Programming Contest

Bucharest, Romania

October 21, 2000

Problem B

Company

Input File: B.DAT

Program Source File: B.PAS or B.C or B.CPP

A company pays different hourly wages for different job types. Each week the company keeps evidence of the total number of work hours for every job type and the total amount paid to all employees for that week. In different weeks different job types can be accomplished. The hourly wage for any job type in the same company remains unchanged. The hourly wage for any job type is a positive integer and the ratio between the maximum wage and the minimum wage is less than 6.

You are asked to write a program that computes the hourly wage for every job type using the data collected a period of time (one or several weeks). The number of job types is limited to 200 and the number of weeks in one data set is limited to 50.

The input file contains several data sets in text format.

The format for the data set is:

- The number of lines for the data set;
- information for one or more weeks

The format of the information for a week is the following:

- job_type1 number_of_hours1
- job_type2 number_of_hours2
- job_type3 number_of_hours3
-
- • total_paid (the dot marks the ending of info for a week)

The job type is represented as a string of characters (limited to 20). The number of hours is a positive integer (smaller than 1E5). The total paid is a positive integer (smaller than 2E10).

The program should write to standard output (for every job type involved in the data set) a line containing the job type and the hourly wage. The output for a data set ends with a line containing a dot.

If the program cannot compute a unique hourly wage for every job type it will print "Incomplete data" and if it cannot compute an integer hourly wage it will print "Inconsistent data"

An example is given in the following table:

input	output
5	Incomplete data
job1 6	.
job2 5	job3 20
job8 4	job2 10
job10 3	job1 30
. 100	.
13	
job3 1	
job2 2	
. 40	
job1 3	
job2 1	
. 100	
job1 1	
job3 2	

job2	3	
.	100	
job1	1	
job2	5	
.	80	



Southeastern European Regional Programming Contest

Bucharest, Romania

October 21, 2000

Problem C MULTIPLE

Input File: C.DAT

Program Source File: C.PAS or C.C or C.CPP

Write a program that, given a natural number N between 0 and 4999 (inclusively), and m distinct decimal digits x_1, x_2, \dots, x_m (at least one), finds the **smallest strictly positive multiple** of N that has no other digits besides x_1, x_2, \dots, x_m (if such a multiple exists).

The input file has several data sets separated by an empty line, each data set having the following format:

- On the first line - the number N
- On the second line - the number m
- On the following M lines - the digits x_1, x_2, \dots, x_m .

For each data set, the program should write to standard output on a single line the multiple, if such a multiple exists, and 0 otherwise.

An example of input and output:

input	output
22	110
3	0
7	
0	
1	
2	
1	
1	



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_identifier2 student_identifier3 .....
```

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	5
0: (3) 4 5 6	2
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	

Figure 1



Problem E
Symbolic Derivation

Input File: E.DAT

Program Source File: E.PAS or E.C or E.CPP

Write a program that performs symbolic derivation $f'(x) = df(x)/dx$ of a given function $f(x)$. The function $f(x)$ is defined by an expression which may contain the following operations: + (addition), - (subtraction), * (multiplication), / (division), and ln (natural logarithm). Besides, the operands may be the variable x and numerical constants. The expression may contain arbitrarily nested sub-expressions in parentheses (). The expression is given in a usual, infix form, such as:

$$(2*\ln(x+1.7)-x*x)/((-7)+3.2*x*x)+(x+3*x)*x$$

Numerical constants have the form $d.d$, with an optional sign (+ or -), where the number of digits both in integer and decimal parts are arbitrary. The input expression is guaranteed to be correct (no syntax error can occur).

The output expression should be written in infix form. It *should not* be optimized for human reading. This means, it can contain redundancies, such as $0*x$, $1*x$, $0+x$, etc. The derivation should be performed using the following rules:

1. The operators * and / are of higher priority than the operators + and -. Parentheses may change the priorities as usually.

2. The operators +, -, *, and / are left-associative, meaning that they group from left to right: $a*b*c = (a*b)*c$, $a/b/c = (a/b)/c$, $a/b*c = (a/b)*c$, etc.

3. The rules for derivation are:

$$(a + b)' = a' + b'$$

$$(a - b)' = a' - b'$$

$$(a * b)' = (a' * b + a * b')$$

$$(a / b)' = (a' * b - a * b') / b^2$$

Note: use b^2 and not $(b*b)$ for presentation

$$\ln(a)' = (a')/a$$

$$x' = 1$$

$$\text{const}' = 0$$

4. While producing the symbolic derivation, use parentheses for output strictly as stated in the previous rule. Do not perform presentation optimizations, such as $0*a = 0$, $1*a = a$, etc.

The input is a textual file which has one $f(x)$ definition per line. The input lines do not have blanks. The output should contain lines with corresponding symbolic derivations $f' = df/dx$, one line for each f . The strings representing $f(x)$ and $f'(x)$ are guaranteed to have no more than 100 characters.

Sample input and output:

Sample Input	Sample Output
$x*x/x$	$((1*x+x*1)*x-x*x*1)/x^2$
$-45.78*x+x$	$(0*x-45.78*1)+1$
$-2.45*x*x+\ln(x-3)$	$((0*x-2.45*1)*x-2.45*x*1)+(1-0)/(x-3)$



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
xmin1 xmax1 ymin1 ymax1
xmin2 xmax2 ymin2 ymax2
...
xminn xmaxn yminn ymaxn
```

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	0 4
4 10 20 10 20 10 20 10 20 10 20 10 20 10 20 10 20	

Figure 1



Problem G
COURSES

Input File: G.DAT

Program Source File: G.PAS or G.C or G.CPP

Consider a group of **N** students and **P** courses. Each student visits zero, one or more than one courses. Your task is to determine whether it is possible to form a committee of exactly **P** students that satisfies simultaneously the conditions:

- every student in the committee represents a different course (a student can represent a course if he/she visits that course)
- each course has a representative in the committee

Your program should read sets of data from a text file. The first line of the input file contains the number of the data sets. Each data set is presented in the following format:

```

P N
Count1 Student1,1 Student1,2 ..... Student1,Count1
Count2 Student2,1 Student2,2 ..... Student2,Count2
.....
CountP StudentP,1 StudentP,2 ..... StudentP,CountP

```

The first line in each data set contains two positive integers separated by one blank: **P** (**1=P=100**) - the number of courses and **N** (**1=N=300**) - the number of students. The next **P** lines describe in sequence of the courses – from *course 1* to *course P*, each line describing a course. The description of *course i* is a line that starts with an integer **Count_i** ($0 \leq \mathbf{Count}_i \leq N$) representing the number of students visiting *course i*. Next, after a blank, you'll find the **Count_i** students, visiting the course, each two consecutive separated by one blank. Students are numbered with the positive integers from 1 to N.

There are no blank lines between consecutive sets of data. Input data are correct.

The result of the program is on the standard output. For each input data set the program prints on a single line "YES" if it is possible to form a committee and "NO" otherwise. There should not be any leading blanks at the start of the line.

An example of program input and output:

input	output
2	YES
3 3	NO
3 1 2 3	
2 1 2	
1 1	
3 3	
2 1 3	
2 1 3	
1 1	



Problem H Closest Common Ancestors

Input File: H.DAT

Program Source File: H.PAS or H.C or H.CPP

Write a program that takes as input a rooted tree and a list of pairs of vertices. For each pair (u, v) the program determines the closest common ancestor of u and v in the tree. The closest common ancestor of two nodes u and v is the node w that is an ancestor of both u and v and has the greatest depth in the tree. A node can be its own ancestor (for example in Figure 1 the ancestors of node 2 are 2 and 5)

The data set, which is read from a text file, starts with the tree description, in the form:

```
nr_of_vertices
vertex:(nr_of_successors) successor1 successor2 ... successorn
...
```

where vertices are represented as integers from 1 to n . The tree description is followed by a list of pairs of vertices, in the form:

```
nr_of_pairs
(u v) (x y) ...
```

The input file contents several data sets (at least one).

Note that white-spaces (tabs, spaces and line breaks) can be used freely in the input.

For each common ancestor the program prints the ancestor and the number of pair for which it is an ancestor. The results are printed on the standard output on separate lines, in to the ascending order of the vertices, in the format: **ancestor:times**

For example, for the following tree:

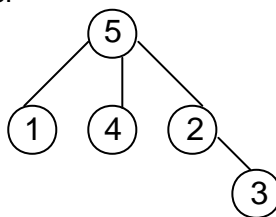


Figure 1

the program input and output is:

Input	Output
5	2:1
5:(3) 1 4 2	5:5
1:(0)	
4:(0)	
2:(1) 3	
3:(0)	
6	
(1 5) (1 4) (4 2)	

(2 3)	
(1 3) (4 3)	