Colombian Collegiate Programming League
CCPL 2014

Contest 12 – October 25

Problems

This set contains 10 problems; pages 1 to 19.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - The Starflyer Agents

Source file name: `agents.c`, `agents.cpp`, *or* `agents.java`

Famed investigator Paula Myo, working on behalf of the 2011 established Commonwealth government, is determined to stop the Starflyer from spying. The Starflyer is a "human-friendly" and powerful alien sentinel intelligence that was found by a space exploration frigate in the Dyson Alpha solar system in year 2285. It is not clear what the Starflyer's real intentions are towards the Commonwealth ... so, it is always better to be safe than sorry!!!

The Starflyer has the ability to control technological equipment; it typically infiltrates droids and uses them as agents. As a matter of fact, droids are carefully identified and tracked in the Commonwealth. Every droid has a history of software updates and each software update is tagged with a hash. A *hash* is a term built recursively from variable, constant, and function symbols as follows:

- any variable and any constant is a hash;
- if each $h_1, \ldots, h_k$ is a hash and f is a function symbol, then $f(h_1, \ldots, h_k)$ is a hash.

As a security measure, a well-kept secret from the general population, the Commonwealth enforces the following policy on droid software updates: for each droid, the tags of any software updates must be compatible. Two hashes h_1 and h_2 are *compatible* if there is a mapping θ from variables to hashes such that $h_1\theta = h_2\theta$, where $h_1\theta$ (resp., $h_2\theta$) denotes the simultaneous replacement of any occurrence of each variable x in h_1 (resp., h_2) with the hash $\theta(x)$. A sequence of hashes $h_1, \ldots, h_n$ is *compatible* if there is θ such that $h_1\theta, \ldots, h_n\theta$ are all equal.

For example, assume that X, Y, Z are variables, c, d are constants, and f, g are function symbols, and consider the hashes h_1, h_2, and h_3 as follows:

$$h_1 : f(X, g(c)) \qquad h_2 : f(f(Y), Z) \qquad h_3 : f(c, g(Y, d))$$

Observe that h_1 and h_2 are compatible because the mapping $\theta = \{X \mapsto f(Y), Z \mapsto g(c)\}$ satisfies $h_1\theta = h_2\theta$. However, any other pair from h_1, h_2, and h_3 is not compatible. Therefore, any sequence of hashes containing h_1, h_2, and h_3 is not compatible because there is no mapping θ such that $h_1\theta = h_2\theta = h_3\theta$.

Detective Myo has just been briefed on the aforementioned security policy. She strongly believes that the Starflyer infiltrates the droids via software updates without having any knowledge of the security policy. If her intuition is right, then this is the chance to detect and stop some Starflyer agents. You have been assigned to Myo's team: your task is to write an algorithm for determining if a sequence of hashes is compatible or not.

Can you help Detective Myo to uncover the Starflyer agents?

Input

The input consists of several test cases. The first line of each test case contains a string $name$ and a natural number n separated by a blank ($2 \leq n \leq 20$, $1 \leq |name| \leq 16$). Then n lines follow, each containing a hash h_i ($1 \leq i \leq n$, $1 \leq |h_i| \leq 512$). You can assume that:

- The $name$ is an alphanumeric text (without blanks) that has a length less than or equal to 16 characters.
- Each one of the n hashes was built according to the above definition and has a length less than or equal to 512 characters.
- The variables, constants, and function symbols are formed exclusively from alphabetic characters. The first character of a variable symbol is an uppercase letter and the first character of a constant or function symbol is a lowercase letter.

The last test case is followed by a line with the text "`END 0`".

The input must be read from standard input.

Output

For each test case, a line must be printed. If the sequence of hashes $h_1, \ldots, h_n$ is compatible, then print the line

```
analysis inconclusive on XXX
```

or if the sequence of hashes $h_1, \ldots, h_n$ is not compatible, then print the line

```
XXX is a Starflyer agent
```

where `XXX` corresponds to $name$ in the test case.

The output must be written to standard output.

Sample Input	**Sample Output**
r2d2 3 f(X,g(c)) f(f(Y),Z) f(c,g(Y,d)) c3po 2 f(X,g(c)) f(f(Y),Z) PC2 2 f(f(Y),Z) f(c,g(Y,d)) END 0	r2d2 is a Starflyer agent analysis inconclusive on c3po PC2 is a Starflyer agent

B - Bob's Beautiful Balls

Source file name: `bob.c`, `bob.cpp`, *or* `bob.java`

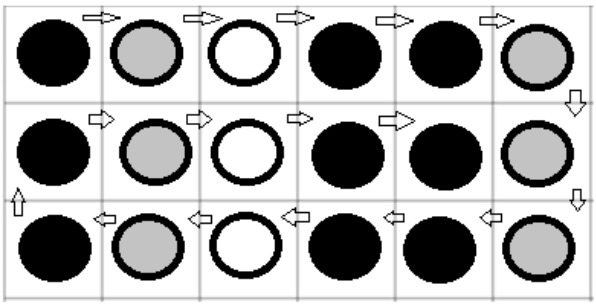
Bob has got B balls and he would like to fill up an $N \times M$ grid with those. Initially he is standing on cell $(1, 1)$, i.e. top left cell, and is facing east. He places the first ball there and moves one cell in the direction he is facing. If he reaches an edge of the grid or if the next move would take him to a cell containing a ball, he turns 90 degrees clockwise before making the next move. Once he lands on an empty cell, he places the next ball there. Since Bob is always concerned with beauty, he wants to see the final grid to have a beautiful configuration.

A grid has a *beautiful configuration* if every column has balls of the same color.

Bob does not like grids with only a single row or a single column – that means both N and M should be greater than one. Since all the cells will finally contain a ball each, the product of N and M should be equal to B. If there are multiple ways of choosing N and M, Bob will choose the pair whose sum $N + M$ is minimized.

Bob has balls of three colors, Black, Grey and White and will be represented by B, G, and W respectively.

An example: suppose Bob has 18 balls in the order `BGWBBGGGBBWGBBGWBB`. He can fill up a 3×6 grid as shown in the following diagram. Note that the placement orders are indicated by arrows. As you can see, all the columns have balls of the same color and thus the grid is beautiful.



Input

The first line of input is an integer that indicates the number of test cases. Each case is a line containing a string that consists of the letters `BGW` only. This ith character of the string gives the color of the ith ball. You can assume the lengths of the strings to be at least 1 and not greater than 100.

The input must be read from standard input.

Output

For each case, output the case number first. If it's possible to form a beautiful grid, output the sum $N + M$ according to the description above. If it's not possible, output -1 instead.

The output must be written to standard output.

Sample Input	**Sample Output**
3 BGWBBGGGBBWGBBGWBB GBBWBBWBBB BBBBBBBBBBBBBB	Case 1: 9 Case 2: -1 Case 3: 8

C - Containers

Source file name: `containers.c`, `containers.cpp`, *or* `containers.java`

A seaport container terminal stores large containers that are eventually loaded on seagoing ships for transport abroad. Containers coming to the terminal by road and rail are stacked at the terminal as they arrive.

Seagoing ships carry large numbers of containers. The time to load a ship depends in part on the locations of its containers. The loading time increases when the containers are not on the top of the stacks, but can be fetched only after removing other containers that are on top of them.

The container terminal needs a plan for stacking containers in order to decrease loading time. The plan must allow each ship to be loaded by accessing only topmost containers on the stacks, and minimizing the total number of stacks needed.

For this problem, we know the order in which ships must be loaded and the order in which containers arrive. Each ship is represented by a capital letter between `A` and `Z` (inclusive), and the ships will be loaded in alphabetical order. Each container is labeled with a capital letter representing the ship onto which it needs to be loaded. There is no limit on the number of containers that can be placed in a single stack.

Input

The input file contains multiple test cases. Each test case consists of a single line containing from 1 to 1000 capital letters representing the order of arrival of a set of containers. For example, the line `ABAC` means consecutive containers arrive to be loaded onto ships `A`, `B`, `A`, and `C`, respectively. When all containers have arrived, the ships are loaded in strictly increasing order: first ship `A`, then ship `B`, and so on.

A line containing the word `end` follows the last test case.

The input must be read from standard input.

Output

For each input case, print the case number (beginning with 1) and the minimum number of stacks needed to store the containers before loading starts. Your output format should be similar to the one shown here.

The output must be written to standard output.

Sample Input	**Sample Output**
A CBACBACBACBACBA CCCCBBBBAAAA ACMICPC end	Case 1: 1 Case 2: 3 Case 3: 1 Case 4: 4

D - A Digital Satire of Digital Age

Source file name: `digital.c`, `digital.cpp`, *or* `digital.java`

The government of "Moderdesh" is planning to enter the digital age and so people of different profession and business are proposing different ways to enter that age successfully. The hardware vendors are saying that we need to provide a laptop for each student, the mobile companies are saying that every children needs to have a mobile phone in his small hand and talk all night long, the multimedia experts are crying for Multimedia University, and so on. But very few are crying for the overall improvement of Computer Science education. We do not understand that by being only the consumer of modern digital technologies we cannot enter the real digital age.

Now as a protest, some local computer geeks are planning to digitize the local vegetable and grocery markets in a strange way. The local markets generally use weighing balances. The computer geeks want to introduce a new type of weight set, where each piece will have the shape of an upper case English alphabet. Conventional weight sets can be related with their ASCII valuess. For example the ASCII value of 'A' is 65, which is 1000001 in binary. For each 1 in binary representation a weight of 500 grs will be added and for each 0 in binary representation a weight of 250 grs will be added. So a piece with shape 'A' actually weighs $250 \times 5 + 2 \times 500 = 2250$ grams. Note that leading zeroes in binary representation are not considered. The geeks believe that if others are correct about their ways to enter digital age, they are also correct about digitizing the local markets by introducing new weight sets related to ASCII characters.

Now in this problem you will be given: (i) the picture of a weighing scale and the weight pieces that it contains in left pan and in the right pan, and (ii) you will also be informed which pan is heavier and which pan is lighter (not necessarily correct). You will have to detect whether the given information is correct or not. If the given information is not correct you will have to rectify the picture and show it in the output.

Input

First line of the input file contains a positive integer that denotes the number of test sets. Each set of input is given in a 7×18 grid. This grid actually contains the plain text description of a weighing scale. Each location of the grid is either a dot `.` (ASCII value 46), a front slash `/` (ASCII Value 47), a back slash `\` (ASCII value 92), an under score `_` (ASCII value 95), a pipe `|` (ASCII value 124), or an upper case English letter (ASCII value 65 to 90). The grid is divided into two equal parts by two vertical lines formed by pipe characters. The left part denotes the status of left pan and right part denotes the status of the right pan. The bottom of the pan is formed by 6 under score characters and the ropes attached to the pans are formed with front slash and back slash. The weights on both pans are placed just above the row that denotes bottom of the pan and they are left justified. There can be a maximum of 6 weights on a single pan. There are three possible vertical positioning of the pans: (i) left pan is low and right pan is high, (ii) both pans are in the middle, and (iii) left pan is high and right pan is low. If weight of both pan is same then they should be in state (ii), if the left pan is heavier then they should be in status (i), and so on. In the input the pans are always in position (i), (ii), or (iii) but that may not be the correct position according to the weights they contain.

A line containing 18 (eighteen) – signs follows each set of input.

The input must be read from standard input.

Output

For each set of input produce two or eight line of output. First line should contain the serial of output. If the pans in the input figure are in correct position according to the weights they contain, then in the second line print "`The figure is correct.`" (without the quotes). If the pans are not in correct position, then print the weighing balance again in a 7×18 grid with the pans in the correct position. Look at the output for sample input for exact formatting.

The output must be written to standard output.

Sample Input

```
4
........||.../\...
........||../..\..
.../\...||./....\.
../..\..||/G.....\
./....\.||\______/
/YQYFU.\||........
\______/||........
------------------
.../\...||........
../..\..||........
./....\.||.../\...
/WCGQG.\||../..\..
\______/||./....\.
........||/OYA...\
........||\______/
------------------
.../\...||........
../..\..||........
./....\.||.../\...
/A.....\||../..\..
\______/||./....\.
........||/A.....\
........||\______/
------------------
........||........
.../\...||.../\...
../..\..||../..\..
./....\.||./....\.
/NQ....\||/FG....\
\______/||\______/
........||........
------------------
```

Sample Output

```
Case 1:
The figure is correct.
Case 2:
........||.../\...
........||../..\..
.../\...||./....\.
../..\..||/OYA...\
./....\.||\______/
/WCGQG.\||........
\______/||........
Case 3:
........||........
.../\...||.../\...
../..\..||../..\..
./....\.||./....\.
/A.....\||/A.....\
\______/||\______/
........||........
Case 4:
The figure is correct.
```

E - Flatland

Source file name: `flatland.c`, `flatland.cpp`, *or* `flatland.java`

Once upon a time there was *Flatland*, a world whose inhabitants believed was a 2D rectangular region. Flatlanders (the people inhabiting Flatland) assumed that if somebody traveled long enough in one direction, he/she would fall down over the edge of Flatland. Animated Caribbean Movies (ACM) plans to produce a film about Flatland. Before the script of the film is approved, ACM wants to become familiar with life as it was in Flatland by simulating how the world could evolve from given initial situations and some conditions determining life and death.

Flatlanders were nomads by nature as they were always traveling: all of them traveled at the same speed rate on straight lines but each individual had its own direction. As you may imagine, if one observed the life of a lonely Flatlander, he/she would eventually reach Flatland's edge and die. Nevertheless, if two Flatlanders collided, then their fate was improved as their directions changed: the resulting directions were as the former ones reflected in a mirror bisecting the angle between the former directions of the crashing inhabitants. So, a Flatlander survived because a collision with another one could change his/her direction.

However, there were bad news when more than two Flatlanders collided in a single crash: in that case all of them died, disappearing right there. Note that some Flatlanders could die at the same time. If this was the case, the last name of the list of deads was remembered as the Last Dead One in that moment (to simplify, we assume they used our modern English alphabet and lexicographic order). The survivors venerated the name of the Last Dead One until a new last dead appeared (and some Flatlander disappeared).

ACM's film begins with a given population in Flatland, where names, positions, and directions of every single individual are known. ACM wants you to help them to determine which would be the name of the last Last Dead One in the whole Flatland's life.

Input

There are NC test cases to solve, $0 < NC < 100$. The first line of the input file has NC. After that, for each testcase, a set of lines:

- the first line contains a number n, the number of Flatlanders in the initial world ($1 \leq n \leq 100$);
- the second line contains two positive integer numbers B and H separated by a space, representing the dimensions of Flatland ($2 \leq B, H \leq 100$). Coordinates in Flatland are points (i, j), with $0 \leq i \leq B$, $0 \leq j \leq H$. Flatland's edges are points with coordinates of the form $(0, j)$, $(i, 0)$, (B, j), or (i, H);
- n lines (one per Flatlander) with four numbers and one string: x, y, d_1, d_2, and $name$ separated by a blank. (x, y) represents the position of the Flatlander ($0 < x < B$, $0 < y < H$, and two Flatlanders cannot start in the same position), (d_1, d_2) represents the direction: (d_1, d_2) is a point on some Flatland's edge, so that the Flatlander is moving towards it; $name$ is a string of

one to 10 alphabetical uppercase characters, which represents the name of the Flatlander. You may assume that every Flatlander has a unique name.

The input must be read from standard input.

Output

For each given case, output one line with the name of the Last Dead One.

The output must be written to standard output.

Sample Input	**Sample Output**
2 2 20 23 1 1 0 0 BOB 3 3 3 0 ALICE 3 20 23 2 2 4 0 ALICE 4 2 2 0 BOB 1 3 0 3 CHARLES	ALICE BOB

F - Alternation Formulae

Source file name: `formulae.c`, `formulae.cpp`, *or* `formulae.java`

For a positive integer n, let $S(n)$ be the string defined by the concatenation of the decimal notations (without leading zeroes!) of $1, 2, \ldots, n$. For instance, $S(11) = 1234567891011$.

An (arithmetic) formula F is an *n-alternation* if it is built inserting in the string $S(n)$ arithmetic operators $+, -$ and parentheses $(,)$. Besides of that, it is required that the used arithmetic operators occur alternately in F.

An n-alternation, being an arithmetic formula, has an integer value. The following are two examples of 11-alternations with the indicated values:

$$1 - (2 + 3) - 4 + 5 - 6 + 7 - 8 + 9 - 1 + 0 - 11 = -13$$
$$-1 + 2 - 3 + 4 - 5 + 6 - 7 + 89 - 1 + 011 = 95$$

Let's consider the following puzzle: given two integers n and m ($n > 0$), decide if there exists an n-alternation F that evaluates to m. From the examples above it is clear that it is possible to build 11-alternations that evaluate to -13 and 95. However, it is easy to see that it is impossible to find a 3-alternation that evaluates to 10.

In order to be precise in the description of the required task, an (arithmetic) *formula* is defined as follows:

- The empty string is not a formula.
- A numeric string, i.e., a string made of digits `0` ... `9`, with at most 5 of them, is a formula.
- If α and β are formulae, then $\alpha + \beta$ and $\alpha - \beta$ are formulae.
- If α is a formula, then $+\alpha$, $-\alpha$ and (α) are formulae.

Input

The input consists of several test cases, each one defined by a line containing two blank-separated integers n and m ($1 \leq n \leq 100$, $-10^7 \leq m \leq 10^7$).

The input must be read from standard input.

Output

For each test case, print a line with the character 'Y' if there exists an n-alternation F that evaluates to m, or with the character 'N', otherwise.

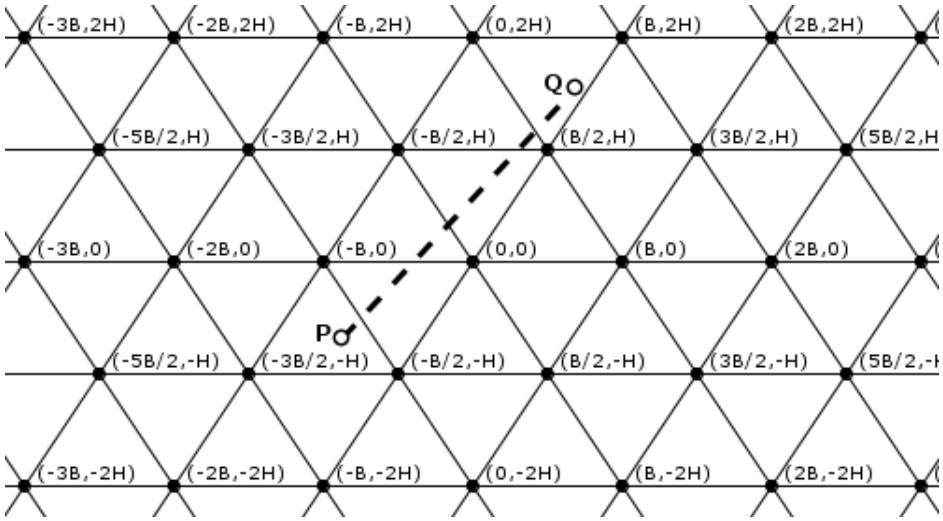
The output must be written to standard output.

Sample Input	**Sample Output**
11 -13 11 95 3 10	Y Y N

G - Triangular Grid

Source file name: `grid.c`, `grid.cpp`, *or* `grid.java`

There is an infinite grid in the Cartesian plane composed of isosceles triangles, with the following design:



A *single triangle* in this grid is a triangle with vertices on intersections of grid lines that has not other triangles inside it.

Given two points P and Q in the Cartesian plane you must determine how many single triangles are intersected by the segment $\overline{PQ}$. A segment intersects a polygon if and only if there exists one point of the segment that lies inside the polygon (excluding its boundary).

Note that the segment $\overline{PQ}$ in the example intersects exactly six single triangles

Input

The problem input consists of several cases, each one defined in a line that contains six integer values B, H, x_1, y_1, x_2 and y_2 ($1 \leq B \leq 200$, $2 \leq H \leq 200$, $-1000 \leq x_1, y_1, x_2, y_2 \leq 1000$), where:

- B is the length of the base of all isosceles single triangles of the grid.
- H is the height of all isosceles single triangles of the grid.
- (x_1, y_1) is the point P, that defines the first extreme of the segment.
- (x_2, y_2) is the point Q, that defines the second extreme of the segment.

You can assume that neither P nor Q lie in the boundary of any single triangle, and that $P \neq Q$.

The end of the input is specified by a line with the string "`0 0 0 0 0 0`".

The input must be read from standard input.

Output

For each case in the input, print one line with the number of single triangles on the grid that are intersected by the segment $\overline{PQ}$.

The output must be written to standard output.

Sample Input	**Sample Output**
100 120 -20 -100 160 160 10 8 5 5 5 4 10 8 5 5 10 5 10 8 5 5 10 10 0 0 0 0 0 0	6 1 2 3

H - Magic Squares

Source file name: `magic.c`, `magic.cpp`, *or* `magic.java`

According to *Wikipedia*, "a *magic square* of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant". This constant is the *module* of the magic square. There are well-known magic squares such as the order 3 chinese Lo Shu magic square:

4	9	2
3	5	7
8	1	6

It is allowed to use any collection of n^2 integer numbers to build a magic square of order n. The Passion façade of the Sagrada Família church in Barcelona, designed by Josep Subirachs, displays the magic square of order 4 and module 33 shown in the following figure. Note that, in this example, the given numbers are not the first n^2 integers and that there are repetitions.

1	14	14	4
11	7	6	9
8	10	10	5
13	2	3	15

Armadora de Cuadrados Magicos (*ACM*) is a recently founded enterprise that is interested on applications of magic squares to cryptography. For that reason, they want to develop software to help magic square builders in detecting if a given sequence of integer numbers may be arranged in a magic square. Your task is to help ACM in this task.

Input

The input consists of several test cases, each one defined by a line containing a sequence of m blank-separated integers $x_1, x_2, \ldots, x_m$ ($1 \leq m \leq 16$, $-10^3 \leq x_i \leq 10^3$ for each $1 \leq i \leq m$).

The input must be read from standard input.

Output

For each test case, output a line with exactly one letter: 'Y' to indicate that a magic square may be built with the numbers provided for the case, or 'N' otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
1 2 3 4 5 6 7 8 9 1 14 4 14 11 7 6 9 8 13 10 2 10 3 5 15 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 8 1 2 3 4 1 1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1	Y Y N N N Y Y N

I - Down Went the Titanic

Source file name: `titanic.c`, `titanic.cpp`, *or* `titanic.java`

After the collision of the great Titanic with the iceberg, it went down. Now there are peoples floating in the cold water struggling with death. Some helping ship will arrive to save them. But they have to survive until the ships arrive. Now consider a water area with people, floating ices, large woods etc. Consider the following symbols:

`*` People staying on floating ice. People want to move from here as the floating ice cannot carry them for long time. Once a people move from here, the floating ice will get drowned. People can move to any of the four directions (north, east, west and south).

$\sim$ Water. People cannot go or move through them as water is extremely cold and not good enough for swimming.

`.` Floating ice. People can move to a floating ice. But floating ices are so light that they cannot float for long time, so people should move from here as soon as possible and once a people move from here, the floating ice will get drowned.

`@` Large iceberg. People can move here but cannot stay here as they are extremely cold. These icebergs will remain floating all the time. Note that, no two people can stay on floating ice or large iceberg at the same time.

`#` Large wood. This place is safe. People can move and stay here until the helping ships arrive. A large wood will get drowned if more than P people stay on it at the same time.

Given the description of the area you have to find an optimal strategy that ensures the maximum number of living people.

Input

The input contains a number of test cases. Each test case starts with a line containing three integers X, Y, and P, where X and Y is the dimensions of the area ($1 \leq X, Y \leq 30$) and P ($P \leq 10$) is the highest capacity of the large woods. Next X lines each contains Y characters. These lines contain no blank spaces or any characters other than asterisk (`*`), tilde ($\sim$), dot (`.`), at (`@`), and hash (`#`). Not more than 50% of the total area has people. Input will terminate with end of input. There is a blank line between two consecutive test cases.

The input must be read from standard input.

Output

For each test case print one line of output, an integer denoting the maximum number of survivors possible.

The output must be written to standard output.

Sample Input	**Sample Output**
3 4 2 *~~# ...@ .~.* 3 5 1 ~~*~~ #.@.# ~~*~~ 1 4 2 **#~	2 2 1

J - The Largest Clique

Source file name: `largest.c`, `largest.cpp`, *or* `largest.java`

Given a directed graph G, consider the following transformation. First, create a new graph $T(G)$ to have the same vertex set as G. Create a directed edge between two vertices u and v in $T(G)$ if and only if there is a path between u and v in G that follows the directed edges only in the forward direction. This graph $T(G)$ is often called the transitive closure of G.

We define a *clique* in a directed graph as a set of vertices U such that for any two vertices u and v in U, there is a directed edge either from u to v or from v to u (or both). The size of a clique is the number of vertices in the clique.

Input

The number of cases is given on the first line of input. Each test case describes a graph G. It begins with a line of two integers n and m, where $0 \leq n \leq 1000$ is the number of vertices of G and $0 \leq m \leq 50000$ is the number of directed edges of G. The vertices of G are numbered from 1 to n. The following m lines contain two distinct integers u and v between 1 and n which define a directed edge from u to v in G.

The input must be read from standard input.

Output

For each test case, output a single integer that is the size of the largest clique in $T(G)$.

The output must be written to standard output.

Sample Input	Sample Output
1 5 5 1 2 2 3 3 1 4 1 5 2	4