	<p>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</p>	<p>13th May 2017 (C, C++, JAVA)</p>
--	---	---

Problem A: In memory of Eric Blow AMAGBEGNON (blue balloon)



Eric Blow AMAGBEGNON was a brilliant, dedicated student who spontaneously made himself available to the ICPC Programming Contest for the organization of the first edition of the National Competition in Ivory Coast. Unfortunately, last weekend, while he was preparing to support his master's thesis, he lost his life.

The Organization Committee has asked you to write a program in his memory.


Given a text, the program must put, all the occurrences of its first and last names, in uppercase. The # character will indicate the end of the text.

Standard Input

The input will contain several lines. Each line will contain no more than 10,000 characters. The text will end with the character '#' on a single line.

Standard Output

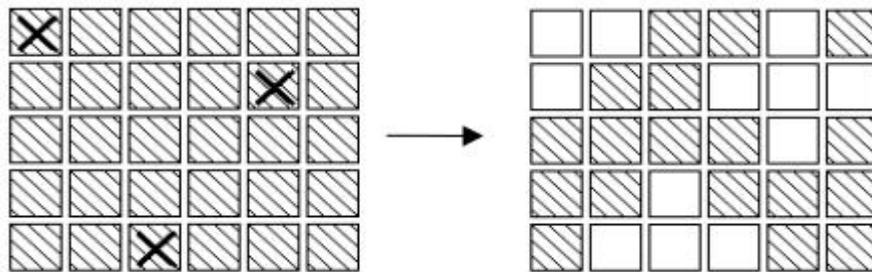
The output will contain the same text with his names in uppercase.

	<p><i>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</i></p>	<p><i>13th May 2017 (C, C++, JAVA)</i></p>
--	--	--

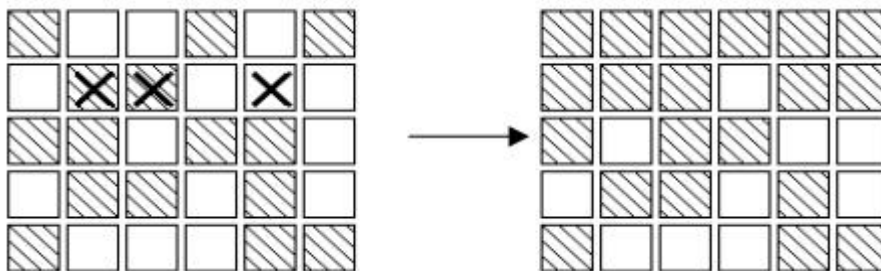
Sample Input	Sample Output
<pre>Eric Blow AMAGBEgNON, the brilliant student, was engaged in ACM programming competition. Eric will always remain in our memory. Thanks you bloW! EricForEver #</pre>	<pre>ERIC BLOW AMAGBEGNON, the brilliant student, was engaged in ACM programming competition. ERIC will always remain in our memory. Thanks you BLOW! EricForEver #</pre>

Problem B: Puzzle (red balloon)

In an extended version of the game *Lights Out*, is a puzzle with 5 rows of 6 buttons each (the actual puzzle has 5 rows of 5 buttons each). Each button has a light. When a button is pressed, that button and each of its (up to four) neighbors above, below, right and left, has the state of its light reversed. (If on, the light is turned off; if off, the light is turned on.) Buttons in the corners change the state of 3 buttons; buttons on an edge change the state of 4 buttons and other buttons change the state of 5. For example, if the buttons marked **X** on the left below were to be pressed, the display would change to the image on the right.




The aim of the game is, starting from any initial set of lights on in the display, to press buttons to get the display to a state where all lights are off. When adjacent buttons are pressed, the action of one button can undo the effect of another. For instance, in the display below, pressing buttons marked X in the left display results in the right display. Note that the buttons in row 2 column 3 and row 2 column 5 both change the state of the button in row 2 column 4, so that, in the end, its state is unchanged.



Note:

1. It does not matter what order the buttons are pressed.
2. If a button is pressed a second time, it exactly cancels the effect of the first press, so no button ever need be pressed more than once.
3. As illustrated in the second diagram, all the lights in the first row may be turned off, by pressing the corresponding buttons in the second row. By repeating this process in each row, all the lights in the first four rows may be turned out. Similarly, by pressing buttons in columns 2, 3 ..., all lights in the first 5 columns may be turned off.

Write a program to solve the puzzle.

	<p>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</p>	<p>13th May 2017 (C, C++, JAVA)</p>
--	---	---


Standard Input

The first line of the input is a positive integer n which is the number of puzzles that follow. Each puzzle will be five lines, each of which has six **0**'s or **1**'s separated by one or more spaces. A **0** indicates that the light is off, while a **1** indicates that the light is on initially.

Standard Output

For each puzzle, the output consists of a line with the string: "**PUZZLE #m**", where m is the index of the puzzle in the standard input. Following that line, is a puzzle-like display (in the same format as the input). In this case, **1**'s indicate buttons that must be pressed to solve the puzzle, while **0**'s indicate buttons, which are not pressed. There should be exactly one space between each **0** or **1** in the output puzzle-like display.

Sample Input	Sample Output
<pre>2 0 1 1 0 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 1 0 0 0 1 0 1 0 0</pre>	<pre>PUZZLE #1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 PUZZLE #2 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 1</pre>

	<p>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</p>	<p>13th May 2017 (C, C++, JAVA)</p>
--	---	---

Problem C: Three (yellow balloon)

Consider the series of numbers whose all prime factors have 3 as their least (right-most) digit. For example, the first 10 numbers in this series are:

3 9 13 23 27 39 43 53 69 73

The numbers **3**, **13**, **23**, **43**, **53**, and **73** are in this series since they're all primes whose least digit is a **3**. Whereas **9** (**3 * 3**), **27** (**3 * 3 * 3**), **39** (**3 * 13**), and **69** (**23 * 3**) are in since all their prime factors have a **3** as their least digit.

Write a program that takes a list of positive integers and determines if each integer is in this series or not.


Standard Input

The standard input contains a list of one or more positive integers, each given on a separate line. Each integer is less than a million. The last line of the standard input contains a -1 (which is not part of the list.)

Standard Output

For each number in the standard input, write, on a separate line, the number itself followed by the word "**YES**" if the number is in the series described above, or "**NO**" if it isn't. Separate the number from the answer by a single space.

Sample Input	Sample Output
3 13 33 -1	3 YES 13 YES 33 NO

	<p>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</p>	<p>13th May 2017 (C, C++, JAVA)</p>
--	---	---

Problem D: Game (green balloon)

Bob has learned a new magic trick that needs a very special preparation. Once he masters the trick he will be able to bring peace to the world, but if he fails, the world will be destroyed. The preparation is performed as follows: There are two containers, initially one is empty and the other one has X marbles. Bob has a Marble Cloning Machine, it clones the marbles in the container with the larger number of marbles, then pours the new clones into the other container (e.g. if the two containers have 7 and 4 marbles, after the cloning step they will have 7 and 11 marbles). The machine does this cloning operation exactly M times. However, there is a bug in the machine, after it performs N cloning operations ($N \leq M$), it will add Y extra marbles to the container with the larger number of marbles. Then the machine will continue normally with the cloning operation exactly $M - N$ times. During the cloning operations, if both containers have the same number of marbles, any of them can be considered the one with the larger number of marbles. Now, the bug in Bob's machine is threatening to destroy the world. But his nerdy friend Alice told him that she knows how to fix it. All he has to do is to calculate the greatest common divisor of the sizes of the two containers after the cloning machine is done. Can you help Bob save the world?

Standard Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T ($1 \leq T \leq 1,000$) representing the number of test cases. Followed by T test cases. Each test case will consist of a single line, containing 4 integers separated by a single space X , N , Y and M ($1 \leq X, Y \leq 1,000$) ($0 \leq N \leq 70$) ($N \leq M \leq 100,000$) which are the numbers as described above.

Standard Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the greatest common divisor of the sizes of the two containers after the machine is done.

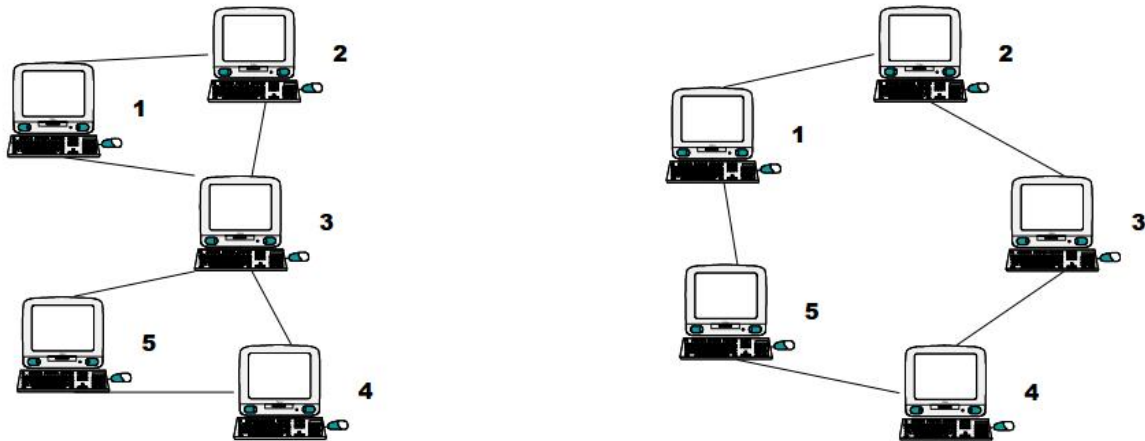
Sample Input	Sample Output
2 4 3 6 5 5 1 15 2	Case 1: 2 Case 2: 5

Note

In the first sample test case, the number of marbles in each container will be the following after each step:

(4, 0), (4, 4), (4, 8), (12, 8), (18, 8), (18, 26), (44, 26). The greatest common divisor of 44 and 26 is 2.

Problem E: Networks (white balloon)



Consider the two networks shown below. Assuming that data moves around these networks only between directly connected nodes on a peer-to-peer basis, a failure of a single node, 3, in the network on the left would prevent some of the still available nodes from communicating with each other. Nodes 1 and 2 could still communicate with each other as could nodes 4 and 5, but communication between any other pairs of nodes would no longer be possible.


Node 3 is therefore a Single Point of Failure (SPF) for this network. Strictly, an SPF will be defined as any node that, if unavailable, would prevent at least one pair of available nodes from being able to communicate on what was previously a fully connected network. Note that the network on the right has no such node; there is no SPF in the network. At least two machines must fail before there are any pairs of available nodes which cannot communicate.

Standard Input

The input will contain the description of several networks. A network description will consist of pairs of integers, one pair per line, that identify connected nodes. Ordering of the pairs is irrelevant; 1 2 and 2 1 specify the same connection. All node numbers will range from 1 to 1000. A line containing a single zero ends the list of connected nodes. A line containing a single zero ends the standard input. Blank lines in the input should be ignored.


Standard Output

For each network in the input, you will output its number, followed by a list of any SPF nodes that exist. The first network in the standard input should be identified as "Network #1", the second as "Network #2", etc. For each SPF node, output a line, formatted as shown in the examples below, that identifies the node and the number of fully connected subnets that remain

	<p><i>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</i></p>	<p><i>13th May 2017 (C, C++, JAVA)</i></p>
--	--	--

when that node fails. If the network has no SPF nodes, simply output the text “No SPF nodes” instead of a list of SPF nodes.

Sample Input	Sample Output
<pre> 1 2 5 4 3 1 3 2 3 4 3 5 0 1 2 2 3 3 4 4 5 5 1 0 1 2 2 3 3 4 4 6 6 3 2 5 5 1 0 0 </pre>	<pre> Network #1 SPF node 3 leaves 2 subnets Network #2 No SPF nodes Network #3 SPF node 2 leaves 2 subnets SPF node 3 leaves 2 subnets </pre>

	<p>The 2017 Collegiate Programming Contest (Benin, Burkina Faso, Ivory Coast, Niger, Senegal, Togo)</p>	<p>13th May 2017 (C, C++, JAVA)</p>
--	---	---

Problem F : Email

(orange balloon)

Marien is the Information System Manager of MapCom Group. MapCom has offices in Benin, Burkina Faso, Ivory Coast, Togo and Niger. For each new employee, he has to generate a new email address. To do that, he uses the first character of the first name and the last name concatenated to « @mapcom-group.com ». For example for Ismael Coulibaly where Ismael is the first name and Coulibaly is the last name, the email address that will be generated is icoulibaly@mapcom-group.com.

In this problem it is assumed that, for each person, only one first name and only one last name will be given in this order.

Your task is to help Marien with a small program that will output email addresses.

Standard Input

The first line of input contains a single integer N , ($1 \leq N \leq 10000$), which is the number of data sets that follow. Each data set should be processed identically and independently. Each data set consists of a single line of input. It contains the first name followed by the last name of one person. Each name will not have more than 256 characters.

Standard Output

For each data set there is one line of output. The single output line consists of the generated email address in uppercase, in a correct output format.

Sample Input	Sample Output
6 Hugues DEGLA Ifede ATCHADE Bienvenu ADANKON Hugues GNIMADI Esse NENONENE Ernest TODEGLA	hdegla@mapcom-group.com iatchade@mapcom-group.com badankon@mapcom-group.com hgnimadi@mapcom-group.com enenonene@mapcom-group.com etodegla@mapcom-group.com