Problem A
Contract revision

For years, all contracts of the Association of Contracts for Modernization (ACM) were typed using an old typewriter machine.

Recently Mr. Miranda, one of the accountants of the ACM, realized that the machine had a failure in one, and only one, numerical digit. More specifically, the flawed digit, when typed, is not printed on the sheet, as if the corresponding key was not pressed. He realized that this could have changed the numerical representation of contract values. Worried about accounting, Mr. Miranda wants to know, from the original values agreed for the contracts (which he kept in handwritten notes) which values are actually represented in the contracts. For example, if the failed digit in the machine is 5, an agreed value of 1500 would be represented in the corresponding contract as 100, because the digit 5 would not be printed. Note that Mr. Miranda wants to know the *numeric value* represented in the contract, ie, in the same machine, the number 5000 corresponds to the numeric value 0, not 000 (as it actually appears in the contract).

Input

The input consists of several test cases, each in one line. Each line contains two integers D and N ($1 \leq D \leq 9$ and $1 \leq N < 10^{100}$), representing, respectively, the digit that has failed in the machine and the number that was originally agreed for the contract (which can be very large because of hiperinflation).

The last test case is followed by a line which contains only two zeros separated by white space.

Output

For each test case in the input your program must print one line containing a single integer, the numeric value represented in the contract.

Sample input	**Sample output**
5 5000000	0
3 123456	12456
9 23454324543423	23454324543423
9 99999999991999999	1
7 777	0
0 0	

Problem B
Sticker Collector Robots

One of the favorite sports in RoboLand is the Robots Rally. This rally is practiced in a giant rectangular arena of square cells with dimensions N rows by M columns. Some cells are empty, some contain a sticker for the World Footbal Cup Album (much appreciated by artificial intelligences in RoboLand) and some are occupied by pillars which support the roof of the arena. During the rally the robots can occupy any cell in the arena, except those containing pillars, since they block the robot movement. The path of the robot in the arena during the rally is determined by a sequence of instructions. Each instruction is represented by one of the following characters: '`D`', '`E`' and '`F`', meaning, respectively, "turn 90 degrees to the right", "turn 90 degrees to the left" and "move forward one cell". Robots start the rally in some initial position in the arena and follow closely the sequence of instructions given (after all, they are robots!). Whenever a robot occupies a cell that contains a Cup sticker he collects it. Stickers are not replaced, that is, each scticker can be collected only once. When a robot tries to move into a cell which contains a pillar he stalls, remaining in the cell where he was, with the same orientation. The same happens when a robot tries to leave the arena.

Given the map of the arena, describing the position of pillars and sctickers, and the sequence of instructions of a robot, you must write a program to determine the number of stickers collected by the robot.

Input

The input contains several test cases. The first line of a test case contains three integers N, M and S ($1 \leq N, M \leq 100$, $1 \leq S \leq 5 \times 10^4$), separated by white spaces, indicating respectively the number of rows, the number of columns of the arena and the number of instructions to the robot. Each of the following N lines describes a cell line of the arena and contains a string with M characters. The first line to appear in the description of the arena is the one more to the North, the first column to appear in description of a cell line of the arena is the one more to the West.

Each cell in the arena is described by one of the following characters:

- '`.`' — normal cell;
- '`*`' — cell which contains a sticker;
- '`#`' — cell which contains a pillar;
- '`N`', '`S`', '`L`', '`O`' — cell where the robot starts the rally (only one in the arena). The letter represents the initial robot orientation (North, South, East and West, respectively).

The last line in the input contains a sequence of S characters among '`D`', '`E`' and '`F`', representing the instructions to the robot.

The last test case is followed by a line which contains only three numbers zero separated by a blank space.

Output

For each rally described in the input your program must print a single line containing a single integer indicating the number of stickers that the robot collected during the rally.

Sample input	**Sample output**
3 3 2 *** *N* *** DE 4 4 5 ...# *#O. *.*. *.#. FFEFF 10 10 20 ....*..... ....*.. .....*.... ..*.#..... ...#N.*..* ...*...... .... FDFFFFFFEEFFFFFFEFDF 0 0 0	0 1 3

Problem C
Account Book

The FCC (Foundation for Combating Corruption) dismantled a major corruption scheme in Nlogonia. During the operation, several account books, with notes documenting the illicit transactions carried out by the scheme, were seized by FCC agents.

Each page on the account books contains some transactions (income or expense, in nilogos, the local currency of Nlogônia, whose symbol is N$) and the cash flow resulting from transactions on that page. For example, if a page recorded the following transactions: an income of N$ 7, an income of N$ 2, an expense of N$ 3, an income of N$ 1 and an expense of N$ 11, the cash flow on that page would be $7 + 2 - 3 + 1 - 11 = -4$.

However, to obstruct the work of the police, the offenders did not record in their account books the type of each transaction (income or expense). In the example above, the page would contain only the numbers 7, 2, 3, 1 and 11 (with no indication whether they were income or expense transactions). The cash flow for each page, however, was always recorded normally, with the signal (in this case, -4).

To guarantee that the offenders are convicted, prosecutors must be able to determine with certainty whether each transaction is an income or an expense. In the example above, transaction N$ 7 was certainly an income, and transaction N$ 11 was certainly an expense. But we cannot say anything about transactions N$ 2, N$ 3, and N$ 1. Transactions N$ 2 and N$ 1 could have been income and in this case transaction N$ 3 would have been an expense; or N$ 1 and N$ 2 could have been expenses and in this case transaction N$ 3 would be an income.

Many account books have a relatively large number of pages, with many transactions, making it is difficult for the police to process all the information. Therefore, they need a program that performs the task efficiently.

Input

The input consists of several test cases. The first line of a test case contains two integers N and F, indicating the number of transactions on the page ($2 \leq N \leq 40$) and cash flow for this page ($-16000 \leq F \leq 16000$). Each of the following N lines contains an integer T_i indicating the value of the i-th transaction ($1 \leq T_i \leq 1000$).

The last test case is followed by a line containing only two numbers zero separated by a space.

Output

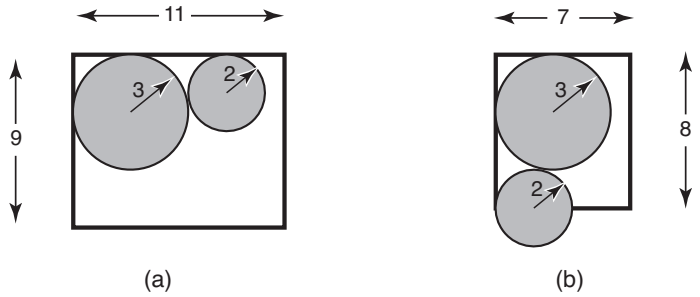
For each test case the input your program must print a single line with N characters. The i-th character must be '`+`', if it is possible determine with certainty that the i-th transaction is an income, '`-`', if it is possible to determine with certainty that the i-th operation is an expense, and '`?`', if it is impossible to determine with certainty the type of transaction. If the cash flow recorded in the page cannot be obtained from the transactions recorded in the page, your program must print a single line containing the character '`*`'.

Sample input	**Sample output**
5 7 1 2 3 4 5 4 15 3 5 7 11 5 12 6 7 7 1 7 0 0	?+??+ * +??-?

Problem E
Elevator

The FCC (Factory of Cylinders of Carbon) manufactures various types of cylinders of carbon. FCC is installed on the tenth floor of a building, and uses the several building's elevators to transport the cylinders. For security, the cylinders must be transported in the upright position, and since they are heavy, at most two cylinders can be transported in a single elevator ride. The elevators have the shape of a parallelepiped and their height is always greater than the height of the cylinders.

To minimize the number of elevator trips to transport the cylinders, the FCC wants, whenever possible, to put two cylinders in the elevator. The figure below illustrates, schematically (top view) a case where this is possible (a), and a case where this is not possible (b):



As there is a very large amount of elevators and types of cylinders, FCC hired you to write a program that, given the dimensions of the elevator and of the two cylinders, determines whether it is possible to put the two cylinders in the elevator.

Input

The input contains several test cases. The first and only line of each test case contains four integers L, C, R_1 and R_2, separated by blanks, indicating the width ($1 \leq L \leq 100$) and the length ($1 \leq C \leq 100$) of the elevator and the radii of the cylinders ($1 \leq R_1, R_2 \leq 100$).

The last test case is followed by a line containing four zeros separated by blanks.

Output

For each test case your program should print a single line with a single character, '`S`' if you can put the two cylinders in the elevator and '`N`' otherwise.

Sample input	**Sample output**

`11 9 2 3`
`7 8 3 2`
`10 15 3 7`
`8 9 3 2`
`0 0 0 0`
	`S`
	`N`
	`N`
	`S`

Problem I
Come and Go

In a certain city there are N intersections connected by one-way and two-way streets. It is a modern city, and several of the streets have tunnels or overpasses. Evidently it must be possible to travel between any two intersections. More precisely given two intersections V and W it must be possible to travel from V to W and from W to V.

Your task is to write a program that reads a description of the city street system and determines whether the requirement of connectedness is satisfied or not.

Input

The input contains several test cases. The first line of a test case contains two integers N and M, separated by a space, indicating the number of intersections ($2 \leq N \leq 2000$) and number of streets ($2 \leq M \leq N(N-1)/2$). The next M lines describe the city street system, with each line describing one street. A street description consists of three integers V, W and P, separated by a blank space, where V and W are distinct identifiers for intersections ($1 \leq V, W \leq N$, $V \neq W$) and P can be 1 or 2; if $P = 1$ the street is one-way, and traffic goes from V to W; if $P = 2$ then the street is two-way and links V and W. A pair of intersections is connected by at most one street.

The last test case is followed by a line that contains only two zero numbers separated by a blank space.

Output

For each test case your program should print a single line containing an integer G, where G is equal to one if the condition of connectedness is satisfied, and G is zero otherwise.

Sample input	**Sample output**
4 5 1 2 1 1 3 2 2 4 1 3 4 1 4 1 2 3 2 1 2 2 1 3 2 3 2 1 2 2 1 3 1 4 2 1 2 2 3 4 2 0 0	1 1 0 0

Problem J
Optical Reader

Professor John decided to apply only multiple-choice tests to his students. In each test, each question will have five alternatives (A, B, C, D and E), and the teacher will distribute one answer sheet for each student. At the end of the test, the answer sheets will be scanned and processed digitally to obtain the test score for each student. Initially, he asked a nephew, who knows computer programming, to write a program to extract the alternatives marked by the students in the answer sheets. The nephew wrote a good piece of software, but cannot finish it because he needs to study for the ICPC contest.

During processing, the answer sheets are scanned in gray levels between 0 (full black) and 255 (total white). After detecting the position for the five rectangles corresponding to each of the alternatives of a question, the software calculates the average pixel gray level for each rectangle, returning an integer value corresponding to each alternative. If the rectangle was filled properly the average value is zero (all black). If the rectangle was left blank the average value is 255 (white total). Thus, ideally, if the average values for each alternative of a question are (255, 0, 255, 255, 255), we know that the student has marked alternative B for that question. However, as answer sheets are processed individually, the average gray level for a completely filled rectangle is not necessarily 0 (may be higher), and the value for a rectangle not filled is not necessarily 255 (may be less). Professor John determined that rectangle average gray levels would be divided into two classes: those with values equal or lower to 127 are considered black and those with values higher than 127 will be considered white.

Obviously, not necessarily all questions of all sheets are marked correctly. It can happen that a student makes a mistake and marks more than one alternative for the same question, or does not mark any alternative. In such cases, the answer to the question should be disregarded.

Professor John now needs a volunteer to write one program that, given the average gray level values of the five rectangles corresponding to the alternatives of a question, determines which alternative was marked, or whether the answer to the question should be disregarded.

Input

The input contains several test cases. The first line of a test case contains an integer N indicating the number of questions in the answer sheet ($1 \leq N \leq 255$). Each of the N following lines describes the response to a question and contains five integers A, B, C, D and E, indicating the values of average gray levels for each alternative ($0 \leq A, B, C, D, E \leq 255$).

The last test case is followed by a line containing only a number zero.

Output

For each test case the input your program must print N lines, each line corresponding to a question. If the answer to the question was correctly filled in the answer sheet, the line should contain the alternative selected (‘`A`’,‘`B`’, ‘`C`’, ‘`D`’ or ‘`E`’). Otherwise, the line should contain the character ‘`*`’ (asterisk).

Sample input	**Sample output**
3 0 255 255 255 255 255 255 255 255 0 255 255 127 255 255 4 200 200 200 0 200 200 1 200 200 1 1 2 3 4 5 255 5 200 130 205 0	A E C D * * B