Colombian Collegiate Programming League
CCPL 2013

Contest 11 -- October 19

Problems

This set contains 10 problems; pages 1 to 19.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Official Twitter account `@CCPL2003`

A - Burguer Time?

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

Everybody knows that along the more important highways there are countless fast food restaurants. One can find easily hamburgers, hot dogs, pizzas, sandwiches ... food everywhere.

Many times the problem isn't to find a restaurant but a drugstore. After a big lunch with fast food it's normal that we need a drugstore because our stomach begins to feel pain.

Given the locations of the restaurants and drugstores on a highway, you want to determine the minimum distance between a restaurant and a drugstore.

Input

The first line of each test case gives an integer L ($1 \leq L \leq 2000000$) indicating the length of the highway.

The second line of each test case contains a string S of length L, showing the positions of the restaurants and drugstores along the highway in the following way:

- The character ''`R`'' represents a place with a restaurant.
- The character ''`D`'' represents a place with a drugstore.
- The character ''`Z`'' represents a place with a restaurant and a drugstore.
- The character ''`.`'' represents an empty place.

You can assume that every test case has at least one restaurant and at least one drugstore.

The end of the input is indicated with $L = 0$.

The input must be read from standard input.

Output

For each case in the input, print one line with the minimum distance between a restaurant and a drugstore.

The output must be written to standard output.

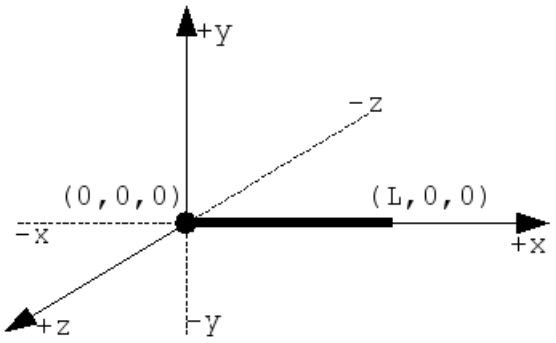
Sample Input	**Sample Output**
2 RD 5 ..Z.. 10 .R......D. 10 .R..Z...D. 10 ...D..R... 25 ..D...R.RR...DD...D.R...R 0	1 0 7 0 3 2

B - Bender B. Rodríguez Problem

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

Bender is a robot built by *Mom's Friendly Robot Company* at its plant in Tijuana, Mexico in 2996. He is a `Bending-Unit 22`, serial number 2716057 and chassis number 1729. He was created for the task of bending metal wires.

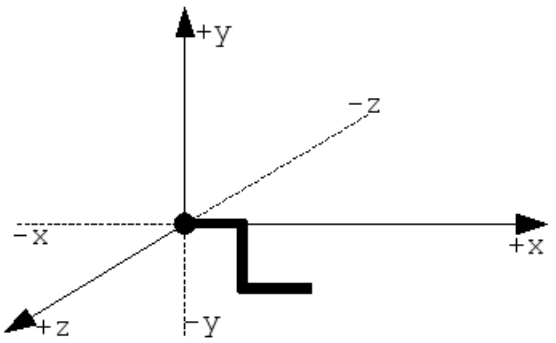
Bender needs to bend a wire of length L ($L \geq 2$ an integer). The wire is represented in the Bender's brain (a `MOS Technology 6502` microprocessor) as a line stucked in the origin of a tridimensional cartesian coordinate system, and extended along the x positive axis ($+x$), so that the fixed extreme of the wire is in the coordinate $(0, 0, 0)$ and the free extreme of the wire is in the coordinate $(L, 0, 0)$.



Bender bends the wire at specific points, starting at the point $(L-1, 0, 0)$ and ending at the point $(1, 0, 0)$. For each i from $L-1$ to 1, Bender can take one of the following decisions:

- Not to bend the wire at point $(i, 0, 0)$.
- To bend the wire at point $(i, 0, 0)$ an angle of $\frac{\pi}{2}$ to be parallel to the axis $+y$, $-y$, $+z$ or $-z$.

For example, if $L = 3$ and Bender bends the wire at $(2, 0, 0)$ on the $+y$ axis direction, and at $(1, 0, 0)$ on the $-y$ axis direction, the result would be:



Given a sequence of bends, you must determine what direction is pointed by the last segment of the wire ($+x$ in the example). You can suppose that the wire can intercept itself, after all it is the future!

Input

The first line of each test case gives an integer L ($2 \leq L \leq 100000$) indicating the length of the wire.

The second line of each test case contains the $L-1$ decisions taken by Bender at each point, separated by spaces. The j-th decision in the list (for each $1 \leq j \leq L-1$) corresponds to the decision taken at the point $(L-j, 0, 0)$, and must be one of the following:

- `No` if the wire isn't bended at point $(L-j, 0, 0)$.
- $+y$ if the wire is bended at point $(L-j, 0, 0)$ on the $+y$ axis.
- $-y$ if the wire is bended at point $(L-j, 0, 0)$ on the $-y$ axis.
- $+z$ if the wire is bended at point $(L-j, 0, 0)$ on the $+z$ axis.
- $-z$ if the wire is bended at point $(L-j, 0, 0)$ on the $-z$ axis.

The end of the input is indicated when $L = 0$.

The input must be read from standard input.

Output

For each case in the input, print one line with the direction pointed by the last segment of the wire, $+x$, $-x$, $+y$, $-y$, $+z$, or $-z$ depending on the case.

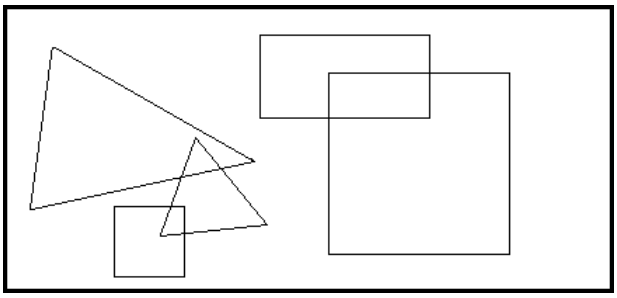
The output must be written to standard output.

Sample Input	Sample Output
3 +z -z 3 +z +y 2 +z 4 +z +y +z 5 No +z No No 0	+x +z +z -x +z

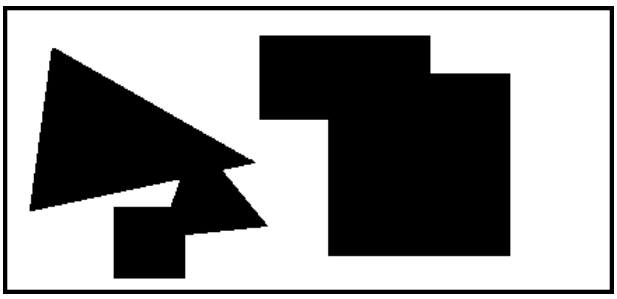
C - Chinese Ink

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

Lucca, my four-years-old daughter, loves drawing polygons in bond paper. For example, yesterday she drew two squares, a rectangle and two triangles:



Today, she wanted to fill her figures with black chinese ink. I helped her, obtaining the next result:



She asked me: how many black zones do you see?. I said: two. I'm bored answering the same question everyday. Can you help us writing a program that, given a collection of black filled polygons, determines the number of black zones on the drawing?

For a precise understanding: a *black zone* is a region of black coloured points on the sheet, where every pair of them may be connected by a continuous line within the region.

Input

The input consists of several test cases. Each test case is represented as follows:

- A line with an integer N ($1 \leq N \leq 40$) which indicates the number of polygons in the drawing.
- N lines, one per polygon, each one containing a list of $2 \cdot t$ integer numbers $x_1\ y_1\ x_2\ y_2 \ldots x_t\ y_t$ specifying the points in the boundary of the polygon ($-10^4 \leq x_i, y_i \leq 10^4$, $3 \leq t \leq 10$). The drawn polygon is bounded by the closed path composed of the straight line segments $\overline{(x_1, y_1)(x_2, y_2)}$, $\overline{(x_2, y_2)(x_3, y_3)}$, ..., $\overline{(x_{t-1}, y_{t-1})(x_t, y_t)}$, and $\overline{(x_t, y_t)(x_1, y_1)}$. You can suppose that the drawn polygon is a simple polygon (a polygon whose boundary is a non-self-intersecting closed path).

The end of the input is indicated when $N = 0$.

The input must be read from standard input.

Output

For each case in the input, print one line with the number of black zones in the drawing after filling each one of the polygons with black chinese ink.

The output must be written to standard output.

Sample Input	**Sample Output**
5 35 29 179 111 19 145 183 22 305 22 305 80 183 80 232 49 361 49 361 178 232 178 137 94 188 156 112 164 79 144 129 143 129 193 79 193 2 20 20 30 20 30 30 20 30 40 40 40 50 50 50 50 40 2 20 20 30 20 30 30 20 30 30 30 40 30 40 40 30 40 3 20 20 40 20 40 40 20 40 50 30 60 20 70 50 60 40 50 30 30 50 0	2 2 1 1

D - Best Coalitions

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

Envy Inc. is a joint stock company, that is, a company in which every stockholder legally owns shares of stock that account for some percentage of the total shares of stock of the company. Due to the global economic crisis, the management rules of Envy Inc. define a particular way for distributing last year's profit: if a stockholder owns more than half of the shares of stock, he/she wins the total profit. Nothing fancy so far in this wild world!

Nevertheless, there are situations in which no stockholder owns more than 50% of the shares of stock of the company. So, in order to gain some profit, stockholders are allowed to form *coalitions*, i.e., groups of stockholders. The participation of the coalition, share-wise, is equivalent to the sum of its stockholders' percentile participation. Hence, if a coalition has more than half of the shares of stock, its members win the totality of last years profit. Then, the members of the coalition receive a part of the profit proportional to their individual participation in the coalition.

For instance, let us assume there are 5 stockholders: A, B, C, D and E, owning 20%, 12%, 14%, 29% and 25% of the stock of the company, respectively. The stockholder E could form several winning coalitions. For example, if E were to form a coalition with A and B, he/she would get 43.86% of last year's profit. If E were to form a coalition with B and C instead, he/she would get 49.02% of last year's profit. On the other hand, E could not form a winning coalition with only A.

Your problem is, given a distribution of shares of stock of Envy Inc., and a stockholder, to determine the maximum percentage of the last year's profit that the given stockholder may win.

Input

The input consists of several test cases, each one defining a percentile distribution of shares of stock, and the index of a stockholder to determine his/her optimal participation. More precisely, each test case is defined by several input lines:

- the first line contains two integer values n ($1 \leq n \leq 100$) and x ($1 \leq x \leq n$), separated by a blank, representing the number of stockholders in Envy Inc. and the index of a stockholder to determine his/her optimal participation, respectively;
- each one of the following n lines has a single floating point value p_i, rounded to 2 decimal places, which represents the percentage of stock ownership of stockholder i ($1 \leq i \leq n$). The floating point delimiter is '`.`' (i.e. the dot). You can assume that $p_1 + \ldots + p_n = 100$.

The end of the input is indicated by $n = x = 0$, an artificial case that must be ignored.

The input must be read from standard input.

Output

For each given case, output a single line with the corresponding answer. The answer should be formatted and approximated to two decimal places. The floating point delimiter must be '.' (i.e. the dot). The rounding applies towards the *nearest neighbor* unless both neighbors are equidistant, in which case the result is rounded up (e.g. 78.312 is rounded to 78.31; 78.566 is rounded to 78.57; 78.345 is rounded to 78.35, etc.).

The output must be written to standard output.

Sample Input	**Sample Output**
5 5 20.00 12.00 29.00 14.00 25.00 2 1 56.87 43.13 2 2 56.87 43.13 3 1 10.00 45.00 45.00 0 0	49.02 100.00 43.13 18.18

E - Look-and-Say Sequences

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

A look-and-say sequence is a sequence of integers, expressed in decimal notation, where each sucessive term is generated by *describing* the previous one.

For instance, if x_1 (the first term of the sequence) is 1, the next term is the description of this term, 11 (''one 1''), which is described by 21 (''two 1's''), which is described by 1211 (''one 2 one 1''), etc.; the series continues 111221, 312211, 13112221, ...

Your problem is to build a program that, given the first term of a look-and-say sequence x_1, calculates the j-th digit of the i-th term, x_i.

Input

Each line in the input corresponds to a test case specified by 3 integer values: x_1, i and j, with $1 \leq x_1 \leq 1000$, $1 \leq i \leq 1000$ and $1 \leq j \leq \min(\lfloor \log_{10}(x_i) + 1 \rfloor, 1000)$.

The end of the input is indicated by a line ''`0 0 0`''.

The input must be read from standard input.

Output

For each test case, the program must output a line with the j-th digit of the term x_i of the look-and-say sequence that starts with the term x_1.

The output must be written to standard output.

Sample Input	**Sample Output**

`1 3 1`
`1 3 2`
`1 7 2`
`123 3 1`
`0 0 0`
	`2`
	`1`
	`3`
	`3`

F - Frieze Patterns

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

John is an architect with a rare interest in math. Recently, he discovered some curious way to describe friezes, those decorative borders constructed from repetitive patterns. Indeed, what John discovered were *frieze patterns*, mathematical objects consisting of an infinite set of numbers arranged in a triangular grid:

```
1   1   1   1   1   1   .
  2   1   3   1   .   .   .
    1   2   2   1   .   .   .
      1   1   1   1   1   1   .
```

A frieze pattern has k ($k \geq 2$) rows, each one consisting of an infinite set of numbers. The first and the last row are exclusively composed by ones. Each set of four adjacent numbers

```
    A
  B   C
    D
```

satisfies the relationship $B * C = A * D + 1$. It is not difficult to see that the complete pattern is determined by the values of the first column and this rule. The example above shows some of the first values of a frieze pattern. As a matter of fact, a frieze pattern all of whose values are integer numbers is called an *integer frieze pattern*.

John has the idea of using integer frieze patterns in his designs (for example, using colors instead of numbers), and he knows, from a given sequence of integer values for the first column, if an integer frieze pattern follows. For this setting, he wants to know the value of a position in an integer frieze pattern, given the values of the first column.

Input

The input consists of several test cases. The first line of each test case is an integer value N, $2 < N \leq 1000$, that specifies the number of rows of the pattern. The second line of each test case corresponds to a pair of integer numbers i, j, with $1 < i < N$ and $1 < j < 10^8$, which specify the row and the position in the row of the value to be reported. The last line contains N values, $v_1, v_2, \ldots, v_N$, with $0 < v_i \leq 1000$ (for each $1 \leq i \leq N$), that correspond to the values of the first element in each row (notice that $v_1 = v_N = 1$). It is guaranteed that every described frieze pattern contains only integer values.

The end of the input is recognized by a number 0 in the place that should correspond to the number of rows of a pattern.

The input must be read from standard input.

Output

Each line of the output corresponds to a test case. The line must include a unique integer value corresponding to the value in the position (*i*, *j*) of the frieze pattern.

The output must be written to standard output.

Sample Input	**Sample Output**
4 2 3 1 2 1 1 6 5 9 1 1 1 1 2 1 0	3 4

G - Gray Inc

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

Gray Inc. is a software fabricant specialized in management of Gray codes. An n-binary code, for $n \geq 1$, is a sequence of words $w_0, w_1, ...$ that includes every possible binary word of n bits. An n-Gray code is an n-binary code such that between any two consecutive words there is only one bit that changes, i.e., they differ at exactly one position. As you might be aware by now, there are many n-Gray codes.

Gray Inc. produces its own particular kind of Gray code in the following way (name G_n the produced n-Gray code, $n \geq 1$):

$$G_n = \begin{cases} \langle 0, 1 \rangle & \text{if } n = 1 \\ (0G_{n-1})\,(1G_{n-1}^R) & \text{if } n \geq 2 \end{cases}$$

The notation defining G_n should be understood as follows:

- bA : appends bit b to every element of the sequence A;
- AB : concatenates sequences A and B;
- A^R : sequence with the elements of sequence A in reversed order.

For instance,

$$\begin{aligned} G_2 &= (0G_1)\,(1G_1^R) \\ &= (0\langle 0, 1\rangle)\,(1\langle 0, 1\rangle^R) \\ &= (0\langle 0, 1\rangle)\,(1\langle 1, 0\rangle) \\ &= \langle 00, 01\rangle\langle 11, 10\rangle \\ &= \langle 00, 01, 11, 10\rangle \end{aligned}$$

and

$$\begin{aligned} G_3 &= (0G_2)\,(1G_2^R) \\ &= (0\langle 00, 01, 11, 10\rangle)\,(1\langle 00, 01, 11, 10\rangle^R) \\ &= (0\langle 00, 01, 11, 10\rangle)\,(1\langle 10, 11, 01, 00\rangle) \\ &= \langle 000, 001, 011, 010\rangle\langle 110, 111, 101, 100\rangle \\ &= \langle 000, 001, 011, 010, 110, 111, 101, 100\rangle \end{aligned}$$

Observe that not only G_n is an n-Gray code, but also a *circular* Gray code as the first word in the sequence may be regarded as the successor of the last one in the sequence.

Gray Inc. wants your help to solve the following problem: given a binary word w in G_n and a natural number m, they want to produce the binary word in the sequence G_n that is m words ahead w in the listing (of course, considering the circular ordering described above). Can you help them?

Input

The problem input consists of several cases, each one defined by a line with an integer m ($0 < m \leq 1000$), and a binary n-word w ($1 \leq n \leq 100$), separated by one blank.

The end of the input is given by a line with $m = 0$ and $w = 0$.

The input must be read from standard input.

Output

For each input case, your solution should output the n-binary word that is m words ahead of w in the listing of G_n.

The output must be written to standard output.

Sample Input	**Sample Output**
1 0 3 0 1 1 1 11 6 011 123 010101010 0 0	1 1 0 10 000 111100100

H - Batman

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

Batman, the superhero, has difficult days fighting villains, and today is one of these days. Batman begins the day checking the list of superpowers that he can use to beat the enemies of Gotham City, and the list of villains that he has to beat during the day.

Each superpower has a name, an attack factor and an energy consumption in calories. Each villain has an alias, a defense factor, and the list of the names of the superpowers that can affect him. To beat a specific villain, Batman must attack him with a superpower that can affect the villain and whose attack factor is greater or equal than the defense factor of the villain. Besides, Batman must use the powers of his list in order (possibly skipping some powers) and have to beat all the villains in the same order of the list. Like everybody else, Batman doesn't have unlimited energy ... he can only spend maximum E calories per day.

Given a list of superpowers and a list of villains, is it possible that Batman beats them all without spending more than E calories?

Input

The input consists of several test cases. Each test case is represented as follows:

- A line with three integers P, V, and E ($0 \leq P, V \leq 1000$, $0 \leq E \leq 10^8$), representing (respectively) the number of superpowers, the number of villains and the maximum amount of calories that Batman can spend per day.
- P lines, one per superpower, containing the following information separated by spaces:
  - An alphanumeric string s with the name of the superpower (case sensitive, without spaces).
  - An integer a ($0 \leq a \leq 10000$) specifying the attack factor of the superpower.
  - An integer c ($0 \leq c \leq 100000$) specifying the energy consumption in calories of the superpower.
- V lines, one per villain, containing the following information separated by spaces:
  - An alphanumeric string s with the alias of the villain (case sensitive, without spaces).
  - An integer d ($0 \leq d \leq 10000$) specifying the defense factor of the villain.
  - A list with the names of the superpowers that can affect the villain, separated with commas. The list will not contain repeated names.

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

Suppose that the superpowers have distinct names and that the villains have distinct aliases.

The maximum length of a name or alias is 100 characters and the minimum length of a name or alias is 1 character.

The input must be read from standard input.

Output

For each test case, print the line ''`Yes`'' if it's possible for Batman to beat all the villains spending *E* or less calories. Otherwise, print the line ''`No`''.

The output must be written to standard output.

Sample Input	**Sample Output**

`4 3 1200`
`A 8000 500`
`B 7000 600`
`C 9000 400`
`D 5000 300`
`Joker 6000 B,A`
`Penguin 8000 B,C,D`
`TwoFace 5000 D,A,B,C`
`4 3 1000`
`A 8000 500`
`B 7000 600`
`C 9000 400`
`D 5000 300`
`Joker 6000 B,A`
`Penguin 8000 B,C,D`
`TwoFace 5000 D,A,B,C`
`0 0 0`
	`Yes`
	`No`

I - Informants

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

The Agency of Counter-intelligence of Macondo, ACM in short, is willing to put to the test the quality of its informants with a simple, yet highly accurate, procedure that evaluates how trustworthy a group of informants is.

The ACM determines the confidence of a group of informants by surveying the confidence among them: informants assert their particular opinions about other ones, even themselves. As a result of the survey, the ACM gathers a set of assertions of the form ''*X* says *Y* is reliable'' or ''*X* says *Y* is not reliable''. If *X* happens to be reliable, the ACM assumes that whatever he or she says, can be interpreted to be true. Otherwise, if *X* is not reliable, his or her opinions may be either true or false. At the end, the ACM qualifies the situation by determining the maximum number of informants that can be reliable according to the surveyed answers.

As an example, let's assume there are four informants *A*, *B*, *C* and *D*, with the following surveyed answers: ''*A* says *B* is reliable but *D* is not'', ''*B* says *C* is not reliable'', and ''*C* says *A* and *D* are reliable''. In this case, it happens that at most two informants are reliable.

Your task is to help the ACM by writing an efficient program that, given the results of the survey, computes the maximum number of informants that may be reliable.

Input

The problem's input has several cases. Each test case begins with a line with two nonnegative integer numbers, i ($0 < i \leq 20$) and a ($0 \leq a \leq 800$), separated by blanks. i is the number of informants and a is the number of answers in the confidence survey. Then, a lines follow, each one with two integer numbers x and y ($1 \leq x \leq i$, $1 \leq |y| \leq i$), separated by blanks. If y is positive, the input line means that ''informant x says informant y is reliable''. If y is negative, the then the line means that ''informant x says informant y is not reliable''.

The end of the input is indicated by a line with two 0 values (an artificial case that should be ignored).

The input must be read from standard input.

Output

For each input case, output in a single line the corresponding answer, i.e., the maximum number of reliable informants according to the answers in the survey.

The output must be written to standard output.

Sample Input	**Sample Output**
4 5 1 2 1 -4 2 -3 3 1 3 4 1 1 1 -1 3 3 1 2 2 3 3 -1 0 0	2 0 2

J - Touring Robot

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

TRobots Inc. fabricates and maintains *touring robots*. Every touring robot visits places in a Cartesian plane scene to accomplish its tasks. A task of a robot is a sequence of at least two points the robot must visit in a tour according to the following rules:

1. the robot starts at the first point of the task, facing the second one;
2. the robot moves in the direction it faces;
3. upon arrival to a point in the sequence, the robot turns counterclockwise an angle α, satisfying $0 \leq \alpha < 2\pi$, until it faces the next point in the sequence (convention: the first point in a sequence is considered the following one for the last point in the sequence);
4. the robot ends at the first point of the sequence, facing the second one.

As a net result of a tour, a robot completes an integer number of turns. TRobots Inc. finds the number of turns important as this number determines when a robot needs maintenance. Your job is to help TRobots Inc. calculating the number of turns a robot completes for a given tour.

Input

The input consists of several cases, each one comprising a set of lines with data that defines a task for a robot. A task for the robot is described by several input lines:

- the first line defines the *size* N of the task, an integer satisfying $1 < N < 1000$ and representing the number of places the robot must visit in the tour;
- each of the following N lines has a pair of integer values, representing the Cartesian coordinates (x, y) of a point to visit in the tour ($-10^6 \leq x, y \leq 10^6$). It is guaranteed that each pair of consecutive points (considering the sequence as a circular list) are different.

The end of the input is given by $N = 0$.

The input must be read from standard input.

Output

For each given case, output one line with the number of turns a robot completes for the given tour.

The output must be written to standard output.

Sample Input	**Sample Output**
3 0 0 3 0 1 0 7 3 4 5 1 -2 -2 -2 2 -1 1 4 -1 1 -2 0	1 4