Colombian Collegiate Programming League

CCPL 2017

Round 4 – April 22

Problems

This set contains 10 problems; pages 1 to 15.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - Stammering Aliens

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

Dr. Ellie Arroway has established contact with an extraterrestrial civilization. However, all efforts to decode their messages have failed so far because, as luck would have it, they have stumbled upon a race of stuttering aliens! Her team has found out that, in every long enough message, the most important words appear repeated a certain number of times as a sequence of consecutive characters, even in the middle of other words. Furthermore, sometimes they use contractions in an obscure manner. For example, if they need to say '`bab`' twice, they might just send the message '`babab`', which has been abbreviated because the second '`b`' of the first word can be reused as the first '`b`' of the second one.

Thus, the message contains possibly overlapping repetitions of the same words over and over again. As a result, Ellie turns to you, S.R. Hadden, for help in identifying the gist of the message.

Given an integer m, and a string s, representing the message, your task is to find the longest substring of s that appears at least m times. For example, in the message '`baaaababababbababbab`', the length 5 word '`babab`' is contained 3 times, namely at positions 5, 7, and 12 (where indices start at zero). No substring appearing 3 or more times is longer (see the first example from the sample input). On the other hand, no substring appears 11 times or more (see example 2).

In case there are several solutions, the substring with the rightmost occurrence is preferred (see example 3).

Input

The input contains several test cases. Each test case consists of a line with an integer m ($m \geq 1$), the minimum number of repetitions, followed by a line containing a string s of length between m and 40000, inclusive. All characters in s are lowercase characters from '`a`' to '`z`'. The last test case is denoted by $m = 0$ and must not be processed.

The input must be read from standard input.

Output

Print one line of output for each test case. If there is no solution, output '`none`'; otherwise, print two integers in a line, separated by a space. The first integer denotes the maximum length of a substring appearing at least m times; the second integer gives the rightmost possible starting position of such a substring.

The output must be written to standard output.

Sample Input	**Sample Output**

`3`
`baaaababababbababbab`
`11`
`baaaababababbababbab`
`3`
`cccccc`
`0`
	`5 12`
	`none`
	`4 2`

B - Candle Box

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

Rita loves her Birthday parties. She is really happy when blowing the candles at the *Happy Birthday's* clap melody. Every year since the age of four she adds her birthday candles (one for every year of age) to a candle box. Her younger daydreaming brother Theo started doing the same at the age of three. Rita and Theo boxes look the same, and so do the candles.

One day Rita decided to count how many candles she had in her box: "No, no, no! I'm younger than that!"

She just realized Theo had thrown some of his birthday candles in her box all these years. Can you help Rita fix the number of candles in her candle box? Given the difference between the ages of Rita and Theo, the number of candles in Rita's box, and the number of candles in Theo's box, find out how many candles Rita needs to remove from her box such that it contains the right number of candles.

Input

The input contains several test cases, each as described next. The first line has one integer D ($1 \leq D \leq 20$) corresponding to the difference between the ages of Rita and Theo. The second line has one integer R ($4 \leq R \leq 1\,000$) corresponding to the number of candles in Rita's box. The third line has one integer T ($0 \leq T \leq 1\,000$) corresponding to the number of candles in Theo's box.

The input must be read from standard input.

Output

For each test case, output an integer representing the number of candles Rita must remove from her box such that it contains the right number of candles.

The output must be written to standard output.

Sample Input	Sample Output
2 26 8 2 26 8	4 4

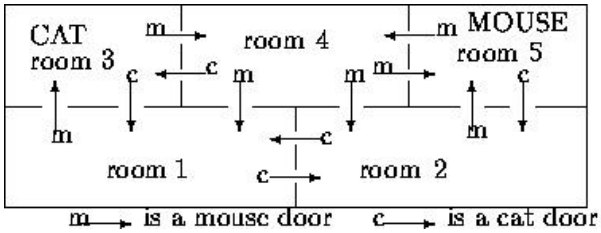
C - Cat and Mouse

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

In a house with many rooms live a cat and a mouse. The cat and the mouse each have chosen one room as their "home". From their "home" they regularly walk through the house. A cat can go from room *A* to room *B* if and only if there is a cat door from room *A* to room *B*. Cat doors can only be used in one direction. Similarly a mouse can go from room *A* to room *B* if and only if there is a mouse door from room *A* to room *B*. Also mouse doors can be used in only one direction. Furthermore, cat doors cannot be used by a mouse, and mouse doors cannot be used by a cat.

Given a map of the house you are asked to write a program that finds out:

1. if there exist walks for the cat and mouse where they meet each other in some room, and
2. if the mouse can make a walk through at least two rooms, end in its "home" room again, and along the way cannot ever meet the cat. (Here, the mouse may not ever meet the cat, whatever the cat does.)



For example, in the map above, the cat can meet the mouse in rooms 1, 2, and 3. Also, the mouse can make a walk through two rooms without ever meeting the cat, e.g., a round trip from room 5 to 4 and back.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input consists of integers and defines the configuration of the house. The first line has three integers separated by blanks: the first integer defines the number of rooms, the second the initial room of the cat (the cat's "home"), and the third integer defines the initial room of the mouse (the mouse's "home"). Next there are zero or more lines, each with two positive integers separated by a blank. These lines are followed by a line with two -1's separated by a blank. The pairs of positive integers define the cat doors. The pair (A, B) represents the presence of a cat door from room A to room B. Finally, there are zero or more lines, each with two positive integers separated by a blank. These pairs of integers define the mouse doors. Here, the pair (A, B) represents the presence of a mouse door from room A to room B.

The number of rooms is at least one and at most 500. All rooms are numbered consecutively starting at 1. You may assume that all positive integers in the input are legal room numbers.

The input must be read from standard input.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output consists of two characters separated by a blank and ended by a new-line character. The first character is 'Y' if there exist walks for the cat and mouse where they meet each other in some room. Otherwise, it is 'N'. The second character is 'Y' if the mouse can make a walk through at least two rooms, end in its "home" room again, and along the way cannot ever meet the cat. Otherwise, it is 'N'.

The output must be written to standard output.

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

D - Gödel's Dream

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

Once Gödel dreamt about codes. And he foresaw a very interesting puzzle involving finite sequences of bits called *h*-sequences. An *h-sequence* is a string defined recursively as follows:

$$\langle hseq \rangle ::= 0$$
$$\langle hseq \rangle ::= 1\langle hseq \rangle\langle hseq \rangle$$

For instance, 0, 11000, and 101100100 are *h*-sequences but 1, 11001, and 111100100 are not.

Gödel's dream was of a tall order of *h*-sequence fun because the strings had incomplete information. More precisely, strings did not only contain bits '0' and '1', but they also contained the question mark symbol '?'. The occurrence of a question mark symbol in a string indicates that any bit can go in such a position of the string. For example, the sequence 101?0 can actually have the *h*-sequence 10100 and the string 10110 (which is not an *h*-sequence) as instances

Given a string made of bits, possibly containing question mark symbols, your task is to write a program that computes the maximum number of *h*-sequences that concatenated together result in an instance of such a string.

Input

The input consists of several test cases, each one defined by a line containing a string *s* made of characters '0', '1', and '?', whose length is between 1 and 10^4 inclusive.

The input must be read from standard input.

Output

For each test case, output a line with the maximum number of *h*-sequences that concatenated together result in an instance of *s*.

The output must be written to standard output.

Sample Input	Sample Output
0 10100 ??1? ?1?? 1?010100 ????1???	1 1 0 2 2 6

E - X-Plosives

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

A secret service developed a new kind of explosive that attain its volatile property only when a specific association of products occurs. Each product is a mix of two different simple compounds, to which we call a *binding pair*. If $N > 2$, then mixing N different binding pairs containing N simple compounds creates a powerful explosive. For example, the binding pairs $A + B$, $B + C$, $A + C$ (three pairs, three compounds) result in an explosive, while $A + B$, $B + C$, $A + D$ (three pairs, four compounds) does not.

You are not a secret agent but only a guy in a delivery agency with one dangerous problem: receive binding pairs in sequential order and place them in a cargo ship. However, you must avoid placing in the same room an explosive association. So, after placing a set of pairs, if you receive one pair that might produce an explosion with some of the pairs already in stock, you must refuse it, otherwise, you must accept it. Compute the number of refusals given a sequence of binding pairs.

Lets assume you receive the following sequence: $A + B$, $G + B$, $D + F$, $A + E$, $E + G$, $F + H$. You would accept the first four pairs but then refuse $E + G$ since it would be possible to make the following explosive with the previous pairs: $A + B$, $G + B$, $A + E$, $E + G$ (4 pairs with 4 simple compounds). Finally, you would accept the last pair, $F + H$.

Input

The input will contain several test cases, each of them as described below. Consecutive test cases are separated by a single blank line. Instead of letters we will use integers to represent compounds. The input contains several lines. Each line (except the last) consists of two integers (each integer lies between 0 and 10^5) separated by a single space, representing a binding pair. Each test case ends in a line with the number '-1'. You may assume that no repeated binding pairs appears in the input.

The input must be read from standard input.

Output

For each test case, output a single line with the number of refusals.

The output must be written to standard output.

Sample Input	Sample Output
1 2 3 4 3 5 3 1 2 3 4 1 2 6 6 5 -1	3

F - Pascal's Hyper-Pyramids

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

We programmers know and love Pascal's triangle: an array of numbers with 1 at the top and whose entries are the sum of the two numbers directly above (except numbers at both ends, which are always 1). For programming this generation rule, the triangle is best represented left-aligned; then the numbers on the left column and on the top row equal 1 and every other is the sum of the numbers immediately above and to its left. The numbers highlighted in bold correspond to the base of Pascal's triangle of height 5:

1	1	1	1	**1**
1	2	3	**4**	
1	3	**6**		
1	**4**			
1				

Pascal's hyper-pyramids generalize the triangle to higher dimensions. In 3 dimensions, the value at position (x, y, z) is the sum of up to three other values:

- $(x, y, z-1)$, the value immediately below it if we are not on the bottom face ($z = 0$)
- $(x, y-1, z)$, the value immediately behind if we are not on the back face ($y = 0$)
- $(x-1, y, z)$, the value immediately to the left if we are not on the leftmost face ($x = 0$);

The following figure depicts Pascal's 3D-pyramid of height 5 as a series of plane cuts obtained by fixing the value of the z coordinate:

$z = 0$

1	1	1	1	**1**
1	2	3	**4**	
1	3	**6**		
1	**4**			
1				

$z = 1$

1	2	3	**4**
2	6	**12**	
3	**12**		
4			

$z = 2$

1	3	**6**
3	**12**	
6		

$z = 3$

1	**4**
4	

$z = 4$

1

For example, the number at position $x = 1$, $y = 2$, $z = 1$ is the sum of the values at $(0, 2, 1)$, $(1, 1, 1)$ and $(1, 2, 0)$, namely, $6 + 3 + 3 = 12$. The base of the pyramid corresponds to a plane of positions such that $x + y + z = 4$ (highlighted in bold above). The size of each layer grows quadratically with the height of the pyramid, but there are many repeated values due to symmetries: numbers at positions that are permutations of one another must be equal. For example, the numbers at positions $(0, 1, 2)$, $(1, 2, 0)$ and $(2, 1, 0)$ above are all equal to 3.

Write a program that, given the number of dimensions D of the hyper-space and the height H of a hyper-pyramid, computes the set of numbers at the base.

Input

The input contains several test cases, each of them as described next. A single line with two positive integers: the number of dimensions D ($2 \leq D < 32$), and the height of the hyperpyramid H ($1 \leq H < 32$). You can assume that D and H are such that all numbers in the hyper-pyramid are less than 2^{63}.

The input must be read from standard input.

Output

For each test case, output the set of numbers at the base of the hyper-pyramid, with no repetitions, one number per line, and in ascending order.

The output must be written to standard output.

Sample Input	**Sample Output**
2 5 3 5	1 4 6 1 4 6 12

G - Performance Review

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

Employee performance reviews are a necessary evil in any company. In a performance review, employees give written feedback about each other on the work done recently. This feedback is passed up to their managers which then decide promotions based on the feedback received.

Maria is in charge of the performance review system in the engineering division of a famous company. The division follows a typical structure. Each employee (except the engineering director) reports to one manager and every employee reports directly or indirectly to the director.

Having the managers assessing the performance of their direct reports has not worked very well. After thorough research, Maria came up with a new performance review system. The main idea is to complement the existing corporate structure with a technical rank for each employee. An employee should give feedback only about subordinates with lower technical level.

Hence, the performance review will work as follows. Employees prepare a summary of their work, estimate how much time it takes to review it, and then request their superiors with higher technical rank to review their work.

Maria is very proud of this new system, but she is unsure if it will be feasible in practice. She wonders how much time each employee will waste writing reviews. Can you help her out? Given the corporate structure of the engineering division, determine how much time each employee will spend writing performance reviews.

Input

The input file contains several test cases, each of them as described next. The first line of input has one integer E ($1 \leq E \leq 10^5$), the number of employees, who are conveniently numbered between 1 and E. The next E lines describe all the employees, starting at employee 1 until employee E. Each line contains three space-separated integers m_i, r_i, t_i ($1 \leq r_i, t_i \leq 10^5$)the manager, the technical rank and the expected time to perform the review of employee i. The engineering director has no manager, represented with $m_i = -1$. The other employees have m_i between 1 and E.

The input must be read from standard input.

Output

For each test case, the output contains E lines. Line i has the time employee i will spend writing reviews.

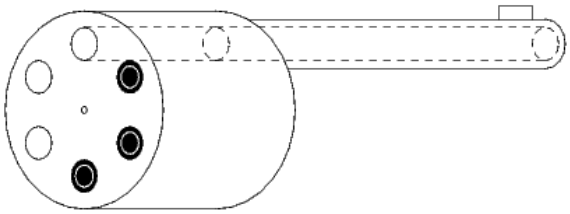
The output must be written to standard output.

Sample Input	Sample Output
5 4 4 80 1 1 40 -1 10 60 3 5 50 4 8 70	40 0 240 120 0

H - Headshot

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

You have a revolver gun with a cylinder that has n chambers. Chambers are located in a circle on a cylinder. Each chamber can be empty or can contain a round. One chamber is aligned with the gun's barrel. When trigger of the gun is pulled, the gun's cylinder rotates, aligning the next chamber with the barrel, hammer strikes the round, making a shot by firing a bullet through the barrel. If the chamber is empty when the hammer strikes it, then there is no shot but just a "click".



You have found a use for this gun. You are playing Russian Roulette with your friend. Your friend loads rounds into some chambers, randomly rotates the cylinder, aligning a random chamber with a gun's barrel, puts the gun to his head and pulls the trigger. You hear "click" and nothing else – the chamber was empty and the gun did not shoot.

Now it is your turn to put the gun to your head and pull the trigger. You have a choice. You can either pull the trigger right away or you can randomly rotate the gun's cylinder and then pull the trigger. What should you choose to maximize the chances of your survival?

Input

The input file contains several datasets. A dataset contains a single line with a string of n binary digits "0" and "1" ($2 \leq n \leq 100$). This line of digits represents the pattern of rounds that were loaded into the gun's chambers. Digit "0" represent an empty chamber and digit "1" represent a loaded one. In this representation, when cylinder rotates before a shot, the next chamber to the right gets aligned with the barrel for a shot. Since the chambers are actually located on a circle, the first chamber in this string follows the last one. There is at least one "0" in this string.

The input must be read from standard input.

Output

For each dataset, write to the output file one of the following words:

`SHOOT` – if pulling the trigger right away makes you less likely to be actually shot in the head with the bullet (more likely that the chamber will be empty).

`ROTATE` – if randomly rotating the cylinder before pulling the trigger makes you less likely to be actually shot in the head with the bullet (more likely that the chamber will be empty).

`EQUAL` – if both of the above choices are equal in terms of probability of being shot.

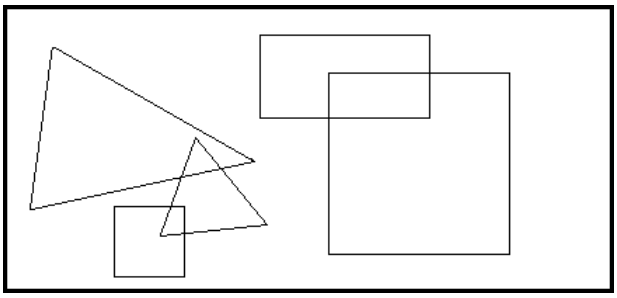
The output must be written to standard output.

Sample Input	**Sample Output**
0011 0111 000111	EQUAL ROTATE SHOOT

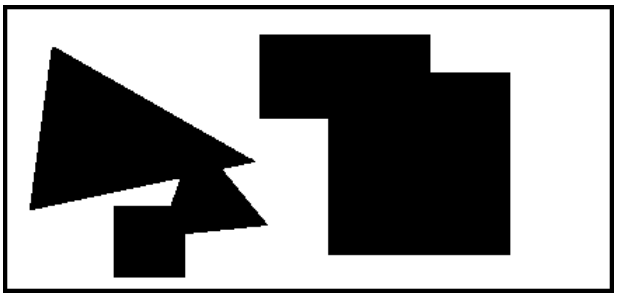
I - Chinese Ink

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

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 $x_1\ y_1\ x_2\ y_2\ ...\ x_t\ y_t$ 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

J - Molar Mass

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

An organic compound is any member of a large class of chemical compounds whose molecules contain carbon. The molar mass of an organic compound is the mass of one mole of the organic compound. The molar mass of an organic compound can be computed from the standard atomic weights of the elements.

When an organic compound is given as a molecular formula, Dr. CHON wants to find its molar mass. A molecular formula, such as $C_3H_4O_3$, identifies each constituent element by its chemical symbol and indicates the number of atoms of each element found in each discrete molecule of that compound. If a molecule contains more than one atom of a particular element, this quantity is indicated using a subscript after the chemical symbol.

In this problem, we assume that the molecular formula is represented by only four elements, C (Carbon), H (Hydrogen), O (Oxygen), and N (Nitrogen). The following table shows that the standard atomic weights for C, H, O, and N.

Atomic Name	Carbon	Hydrogen	Oxygen	Nitrogen
Standard Atomic Weight	$12.01g/mol$	$1.008g/mol$	$16.00g/mol$	$14.01g/mol$

For example, the molar mass of a molecular formula C_6H_5OH is $94.108g/mol$ which is computed by $6 \cdot (12.01g/mol) + 6 \cdot (1.008g/mol) + 1 \cdot (16.00g/mol)$.

Given a molecular formula, write a program to compute the molar mass of the formula.

Input

The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case is given in a single line, which contains a molecular formula as a string. The chemical symbol is given by a capital letter and the length of the string is greater than 0 and less than 80. The quantity number n which is represented after the chemical symbol would be omitted when the number is 1 ($2 \leq n \leq 99$).

The input must be read from standard input.

Output

Print exactly one line for each test case. The line should contain the molar mass of the given molecular formula rounded up to three decimal places.

The output must be written to standard output.

Sample Input	**Sample Output**
4 C C6H5OH NH2CH2COOH C12H22O11	12.010 94.108 75.070 342.296