2015 WVU Tech CSIS Programming Competition

Department of Computer Science and Information Systems
West Virginia University Institute of Technology

April 11, 2015

Instructions

1. There are ten (10) problems in the packet, using letters A-J. These problems are NOT sorted by difficulty.

2. All solutions must read from standard/console input and write to standard/console output.

3. Solutions for problems submitted for judging are called runs. Each run will be judged. Runs for each particular problem will be judged in the order in which they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first. DO NOT request clarifications on when a response will be returned. If you have not received a response for a run within 10 minutes of submitting it, **you may ask the lab monitor to send a runner to the judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
YES	Your submission has been judged correct.
NO - Compile Error	Your submission didn't compile correctly.
NO - Time Limit Exceeded	Your submission ran for longer than 60 seconds without terminating.
NO - Incorrect Answer	Your submission produced an incorrect answer for at least one of the test cases.
NO - Runtime Error	Your submission terminated abnormally.
NO - Other (Please see staff)	Your submission produced an error that is not classified from above. The judge will provide additional information if this happens.

4. A team's score is based on the number of problems they solve and penalty points. The penalty points reflect the time required to solve a problem correctly and the number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charge equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked by the number of problems solved. Ties are resolved in favor of the team with the fewest penalty points.

5. Each problem contains sample input and output. However, you may be assured that the judges will test your submission against several other more complex data sets, which will not be revealed. Before submitting your run, you should design other input sets to fully test your program. Should you receive an incorrect judgment, you are advised to consider what other data sets you could design to further evaluate your program.

6. In the event that your think a problem is ambiguous, you may request a clarification. Read the problem carefully before submitting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient. No clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

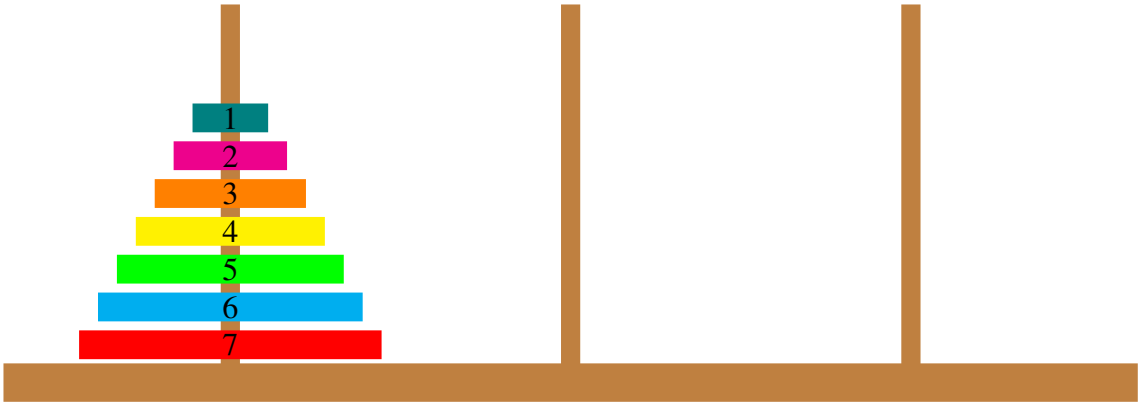
7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

8. Good luck and have fun!

Problem A

Towers of Hanoi

This game consists of three pegs and a number of disks with increasing diameter. An example of this game is shown below, where we are given 7 disks and 3 pegs. The goal is to move the disks from the starting peg to the ending peg without placing a larger disk on top of a smaller disk.



You are given the number of disks to use, the starting peg number, and the ending peg number. For example, `7 1 3` describes 7 disks starting on peg 1 and the goal is to place the disks on peg 3.

Now that you have an idea of the game, here is the problem.
Given the input $n\ s\ e\ m$, where

- n is the number of disks
- s is the starting peg number
- e is the ending peg number
- m is the number of moves

display the configuration of the disks on the pegs after m moves have taken place. The configuration consists of an $n \times e$ grid. The pegs are assumed to be ordered from left to right ($1 \rightarrow 3$). Each disk is a number between 1 and n. For example, if $n = 7$, the disks are numbered 1 to 7. The larger the number, the larger the diameter. A blank is represented on the grid by using 0.

For example, if the input is 7 1 3 5, the following output would be expected:

```
0 0 0
0 0 0
1 0 0
4 0 0
5 0 0
6 0 0
7 2 3
```

Input

Each test case is described by one line of input having the format: $n\ s\ e\ m$, where $1 \leq n \leq 50$, $1 \leq s \leq 3$, $1 \leq e \leq 3$, and $1 \leq m \leq 5000$. Each of these fields will be separated by a space. A test case of $0\ 0\ 0\ 0$ means there are no more test cases.

Output

For each case, display the case number followed by the configuration of the disks and pegs after m moves have taken place.

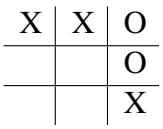
Sample Input	Sample Output
7 1 3 5 4 2 1 6 3 2 3 5 0 0 0 0	Case 1: 0 0 0 0 0 0 1 0 0 4 0 0 5 0 0 6 0 0 7 2 3 Case 2: 0 0 0 0 0 0 0 1 2 0 4 3 Case 3: 0 0 0 0 0 0 2 1 3

Problem B

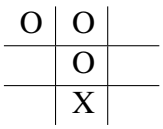
Tic Tac Toe

Given players X and O, the computer will determine who can win the game and the location of a winning move.

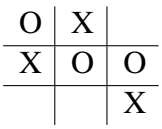
Suppose we are given the following games:



Game 1



Game 2



Game 3

In Game 1, X can win at row 2, column 2. In Game 2, O can win at row 1, column 3. In Game 3, there is no winning move for either player.

Input

Each test case is described by three lines of input representing a Tic Tac Toe board. Each row will contain three characters with no spaces in between: Xs, Os, and Bs, where B is a blank space. End of input is signaled by a line containing 3 zeros.

Output

For each case, display the case number followed by which player can win and where. If there is not a winning move, output the statement "No winning move."

Sample Input	Sample Output
XXO BBO BBX OOB BOB BXB OXB XOO BBX 000	Case 1: X can win at row 2, column 2. Case 2: O can win at row 1, column 3. Case 3: No winning move.

Problem C

Gale-Berlekamp Switching Game

Your room is lit by an $n \times n$ array of light bulbs. You can turn on/off each light bulb using an individual switch that independently controls each light bulb. Your landlord has $2n$ switches, where each switch turns on/off an entire row or column of light bulbs. The landlord does not have access to individual switches, and you do not have access to switches that control entire rows or columns of light bulbs. Your goal is to keep as many light bulbs lit as possible, while the landlord attempts to minimize the number of light bulbs that are on.

We can represent turning on and off light bulbs using a two-dimensional array, where 1 represents a light bulb that is turned on, and 0 represents a light bulb that is turned off.

Suppose $n = 2$, meaning we have a 2×2 array. If the initial configuration is

$$\begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix},$$

the landlord can switch off all light bulbs switching off the first row and then the second row. If the initial configuration is

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix},$$

the landlord can switch the first row to get the configuration

$$\begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix}$$

and then switch the second column to get the configuration

$$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}.$$

However, if the initial configuration is

$$\begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix},$$

the landlord cannot reduce the number of light bulbs that are turned on to zero. It is easy to see that when $n = 2$, the maximum number of light bulbs we can keep on is 1.

Suppose $n = 3$, meaning we have a 3×3 array. If we have the initial configuration

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix},$$

the landlord will throw the switches for the first row and the second row to get the configuration

$$\begin{matrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{matrix}$$

and then use the third column switch to reduce the configuration to

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}.$$

It can be shown that if $n = 3$, then the maximum number of light bulbs that can be kept on is 2.

For any $n \times n$ array of light bulbs, let $R(n)$ denote the maximum number of light bulbs that can be kept on so that the landlord cannot reduce the number to anything smaller. The values for $R(n)$ are given, where $2 \leq n \leq 10$:

n	2	3	4	5	6	7	8	9	10
$R(n)$	1	2	4	7	11	16	22	27	35

Write a program that finds an $n \times n$ configuration of light bulbs, where $2 \leq n \leq 6$ and there are exactly $R(n)$ light bulbs that are turned on. The landlord should not be able to reduce the number of light bulbs lit with this configuration. In other words, the number of ones in this array cannot be reduced by complementing (i.e., changing 1s to 0s and vice versa) any combination of rows and columns.

Input

The input consists of a single integer n, where $2 \leq n \leq 6$. If $n = 0$, then the program should terminate.

Output

The output should consists of n lines, where each line contains n numbers that can be either 0 or 1. The total number of 1s among all rows must be exactly $R(n)$. The output represents a configuration that cannot be further reduced in terms of the number of 1s present.

Note that there is more than one correct answer for a given n. The output simply needs to produce one correct configuration. The sample input/output below provides a few possible solutions. Also, we will check your code to make sure your program is not simply outputting a specific configuration for each n.

Sample Input	Sample Output
3 2 0	0 1 0 1 0 0 0 0 0 1 0 0 0

Problem D

Morse Code

Remington the polar bear and Sally the penguin want to pass encrypted messages via radio signals. Remington lives at the North Pole, and Sally lives in the South Pole. Thus, postage would be outrageously expensive, and it would take a long time to send letters.

Morse code consists of two signals: short and long. These are represented as dots (i.e., `.`) and dashes (i.e., −). Both Remington and Sally will be using the following Morse code encoding:

Letter	Morse Code	Letter	Morse Code	Digit	Morse Code
A	.−.....	a	.−−....−	0	..−−....
B	.−...−.	b	.−−...−.	1	..−−...−
C	.−....−−	c	.−−...−−	2	..−−..−.
D	.−...−..	d	.−−..−..	3	..−−..−−
E	.−...−.−	e	.−−..−.−	4	..−−.−..
F	.−...−−.	f	.−−..−−.	5	..−−.−.−
G	.−...−−−	g	.−−..−−−	6	..−−.−−.
H	.−..−...	h	.−−.−...	7	..−−.−−−
I	.−..−..−	i	.−−.−..−	8	..−−−...
J	.−..−.−.	j	.−−.−.−.	9	..−−−..−
K	.−..−.−−	k	.−−.−.−−	Punctuation	
L	.−..−−..	l	.−−.−−..	Period (.)	..−.−−−.
M	.−..−−.−	m	.−−.−−.−	Comma (,)	..−.−−..
N	.−..−−−.	n	.−−.−−−.	!	..−.−..−
O	.−..−−−−	o	.−−.−−−−	"	..−.−.−.
P	.−.−....	p	.−−−....	#	..−.−.−−
Q	.−.−...−	q	.−−−...−	$	..−..−..
R	.−.−..−.	r	.−−−..−.	%	..−..−.−
S	.−.−..−−	s	.−−−..−−	&	..−..−−.
T	.−.−.−..	t	.−−−.−..	'	..−..−−−
U	.−.−.−.−	u	.−−−.−.−	Hyphen (−)	..−.−−.−
V	.−.−.−−.	v	.−−−.−−.	Colon (:)	..−−−.−.
W	.−.−.−−−	w	.−−−.−−−	;	..−−−.−−
X	.−.−−...	x	.−−−−...	Space	..−....
Y	.−.−−..−	y	.−−−−..−	?	..−−−−−−
Z	.−.−−.−.	z	.−−−−.−.		

The message is encrypted (and decrypted) using simple XOR techniques. Each message starts with a single character that serves as the key of the message. We apply the XOR operation with the key and each remaining character in the message using the dot and dash sequence. The following table provides the results of the XOR operation for dots and dashes:

First Symbol	Second Symbol	XOR
Dot (.)	Dot (.)	Dash (−)
Dot (.)	Dash (−)	Dot (.)
Dash (−)	Dot (.)	Dot (.)
Dash (−)	Dash (−)	Dash (−)

You are asked to write a program that reads in a message (either encrypted or decrypted) and outputs the translated message.

Input

The first line of input consists of either `00` or `99`. `00` means that a message is to be read, while `99` terminates the program.

The first character of the message is the key. Note that the key will always be a letter (either uppercase or lowercase). Anything after the key is the message. The message ends with the word `STOP` on a separate line. Note that the message can have up to 10 lines.

The message can be either encrypted or decrypted. If the message is already encrypted, note that the key is also encrypted in Morse code format.

Output

If the input message was decoded, the input message should be encoded. Note that the key must be included at the beginning in Morse code format. If the input message was encoded, the output message should be decoded. Note that the decoded message should not have the key in front of it.

Sample Input

```
00
RHey Sally! How are you?
STOP
00
SGreetings Remington, Happy Birthday.
STOP
99
```

Sample Output

```
.-.-..-.---..-.---..-...--.-.-..-...--.--------.--..--..
--.....---.....---.-.-..-...--..-...--.----..-.---....-.
--.--.-.-...--.---..--..--.-------..-...-...--.---.-.-..
--....-.--.--...-..-..-.

.-.-..-----.-.----.----.--..-..---..-..---.--...--...-.-
--....-.--..-.----.------...--..-------.--..-..---.....-
--...-.---....-.--..-.----.--...--....----....-.-.......
-...--..---..-.-..--..--.---.---..--.---..--.-.-.--...--..
---.---.--...-.---.----.--.--...--...-..--..-...--..--.-
--.-.-.--.....-.
```

Problem E

Expression Evaluation

Evaluate basic mathematical expressions provided by the user via the command line. The program should read in the user input and output the answer to the mathematical expression. The expressions are limited to the following operators:

- Addition ($+$)
- Subtraction ($-$)
- Multiplication ($*$)
- Division ($/$)

Additionally, there are two more challenges. The use of parentheses is allowed and denotes the order of operations. The mathematical expression can contain spaces but are ignored when evaluating.

Input

Each test case is described by a valid mathematical expression using the above operators. End of input is signaled by a line containing the word "exit."

Output

For each case, display the answer to the mathematical expression.

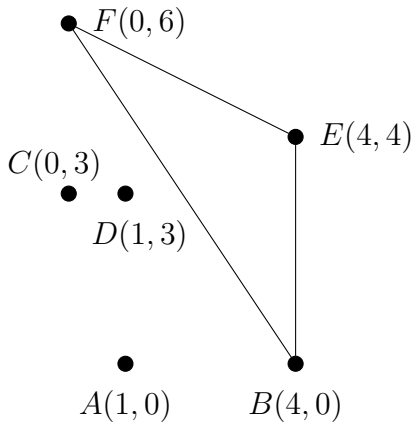
Sample Input	Sample Output

`(4 + 2)`
`3 * (9 - 2)`
`(2 + 4) * (3 - 1)`
`45 - 3 + 0 * 1`
`exit`
	`6`
	`21`
	`12`
	`42`

Problem F

Myacm Triangles

There has been considerable archeological work on the ancient Myacm culture. Many artifacts have been found in what have been called power fields: a fairly small area, less than 100 meters square where there are from four to fifteen tall monuments with crystals on top. Such an area is mapped out below.



Most of the artifacts discovered have come from inside a triangular area between just three of the monuments, now called the power triangle. After considerable analysis, archeologists agree how this triangle is selected from all the triangles with three monuments as vertices: It is the triangle with the largest possible area that does not contain any other monuments inside the triangle or on an edge of the triangle. Each field contains only one such triangle.

Archeological teams are continuing to find more power fields. They would like to automate the task of locating the power triangles in power fields. Write a program that takes the positions of the monuments in any number of power fields as input and determines the power triangle for each power field.

To assist with this computation, you will need to know the area of a triangle with vertices (x_1, y_1), (x_2, y_2), and (x_3, y_3), which is

$$|0.5 \times [(y_3 - y_1)(x_2 - x_1) - (y_2 - y_1)(x_3 - x_1)]|$$

Input

For each power field, there are several lines of data. The first line is the number of monuments: at least 4, and at most 15. For each monument, there is a data line that starts with one character label for the monument and is followed by the coordinates of the monument, which are non-negative integers less than 100. The first label is A, and the next label is B, and so on.

There is at least one such power field described. The end of input is indicated by a 0 for the number of monuments. The first sample data below corresponds to the diagram in the problem.

Output

For each power field, there is one line of output. It contains the three labels of the vertices of the power triangle, listed in increasing alphabetical order, with no spaces.

Sample Input	Sample Output
6 A 1 0 B 4 0 C 0 3 D 1 3 E 4 4 F 0 6 4 A 0 0 B 1 0 C 99 0 D 99 99 0	BEF BCD

Problem G

Orders

The store manager has sorted all kinds of goods in alphabetical order of their labels. All the goods having labels starting with the same letter are stored in the same warehouse (i.e., in the same building) that are labeled with this letter. During the day, the store manager receives and books the orders of goods which are to be delivered from the store. Each order requires only one kind of goods. The manager processes the requests in the order of their booking.

You know in advance all the orders that have to be processed by the manager today, but you do not know their booking order. Compute all possible ways to visit the warehouses for the store manager to settle all the demands piece after piece during the day.

Input

Input contains a single line with all labels of the requested goods (in random order). Each kind of goods is represented by the starting letter of its label. Only lowercase letters of the alphabet are used. The number of orders doesn't exceed 200. There may be multiple input lines, but each line is a single request. An input of `0` indicates the end of input.

Output

Output will contain all possible orderings in which the store manager may visit the warehouses. Every warehouse is represented by a single lowercase letter of the alphabet which corresponds to the start letter of the label of the goods. Each ordering of warehouses is written to the output only once on a separate line and all the lines containing orderings have to be sorted in alphabetical order.

Sample Input	Sample Output
bbjd	bbdj bbjd bdbj bdjb bjbd bjdb dbbj dbjb djbb jbdb jbdb jdbb

Problem H

Gold Coins

The king pays his loyal knight in gold coins. On the first day of his service, the knight receives one gold coin. On each of the next two days (the second and third days of service), the knight receives two gold coins. On each of the next three days (the fourth, fifth, and sixth days of service), the knight receives three gold coins. On each of the next four days (the seventh, eighth, ninth, and tenth days of service), the knight receives four gold coins. This pattern of payments will continue indefinitely: After receiving N gold coins on each of N consecutive days, the knight will receive $N + 1$ gold coins on each of the next $N + 1$ consecutive days, where N is any positive integer.

Your program will determine the total number of gold coins paid to the knight in any given number of days, starting from Day 1.

Input

The input contains at least one, but no more than 21 lines. Each line of the input (except the last one) contains data for one test case of the problem, consisting of exactly one integer (in the range 1 to 10000), representing the number of days. The end of the input is signaled by a line containing the number 0.

Output

There is exactly one line of output for each test case. This line contains the number of days from the corresponding line of input, followed by one blank space and the total number of gold coins paid to the knight in the given number of days, starting with Day 1.

Sample Input	Sample Output
10 6 7 11 15 16 100 10000 1000 21 22 0	10 30 6 14 7 18 11 35 15 55 16 61 100 945 10000 942820 1000 29820 21 29 22 98

Problem I

Fake Coin

The "Gold Bar" bank received information from reliable sources that in their last group of N coins, exactly one coin is a fake and differs in weight from the other coins. Note that all other coins are equal in weight. They have only a simple balance available. Using this balance, one is able to determine if the weight of the objects left in the pan is less than, greater than, or equal to the weight of the objects in the right pan.

In order to detect the fake coin, the bank employees numbered all the coins using integers from 1 to N, thus assigning each coin a unique integer identifier. After that, they began to weigh various groups of coins by placing equal numbers of coins in the left pan and in the right pan. The identifiers of coins and the results of the weighings were recorded.

You are to write a program that will help the bank employees determine the identifier of the fake coin using the results of these weighings.

Input

The first line of the input contains two integers N and K, separated by spaces, where N is the number of coins ($2 \leq N \leq 1000$) and K is the number of weighings fulfilled ($1 \leq K \leq 100$). The following $2K$ lines describe all weighings.

Two consecutive lines describe each weighing. The first of these lines starts with a number M ($1 \leq M \leq N/2$), representing the number of coins placed in the left and in the right pans, followed by M identifiers of coins placed in the left pan and M identifiers of coins placed in the right pan. All numbers are separated by spaces.

The second line contains one of the following characters: `'<'`, `'>'`, or `'='`. It represents the result of the weighing:

- `'<'` means that the weight of the coins in the left pan is less than the weight of the coins in the right pan.
- `'>'` means that the weight of the coins in the left pan is greater than the weight of the coins in the right pan.
- `'='` means that the weight of the coins in the left pan is equal to the weight of the coins in the right pan.

The above input will repeat, meaning there may be multiple test cases, until the user enters `0 0` for N and K.

Output

The output is either the identifier of the fake coin. If the results of the current weighings cannot determing the fake coin, the output should be 0.

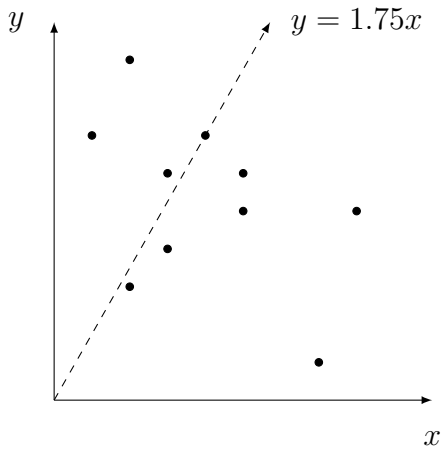
Sample Input	Sample Output
5 3 2 1 2 3 4 < 1 1 4 = 1 2 5 =	3

Problem J

Points Above the Line

You are given a set of coordinates of points that lie on the first quadrant on an xy plane. You are asked to write a program that finds the slope of a line that passes through the origin such that p points are on or above the line.

For example, suppose you are given 10 points with the following coordinates: $(1, 7), (2, 3), (2, 9), (3, 4), (3, 6), (4, 7), (5, 5), (5, 6), (7, 1), (8, 5)$. If $p = 4$, then the slope of the line is 1.75. The diagram below shows these points on the plane and the line.



Input

The first line of input is a single integer that gives the number of points that are on the first quadrant of the plane. We will let n denote this number, where $1 \leq n \leq 1000$. If $n = 0$, then the program should terminate.

The second line is a single integer that tells us the number of points that must be either on or above the desired line that passes through the origin. We will let p denote this number, where $1 \leq p \leq 1000$.

The remaining n lines of input give us the coordinates of the points. Each line has two numbers. The first number is the x-coordinate of the point, and the second number is the y-coordinate of the point. Note that $0 < x, y$, meaning no point will be directly on the x or y-axis. Also, we can assume that no two points lie on the same line passing through the origin.

Output

The output is a single number that gives the slope of the line passing through the origin such that exactly p points are on or above the line. The slope should be rounded two decimal places.

Sample Input	Sample Output
10 4 1 7 2 3 2 9 3 4 3 6 4 7 5 5 5 6 7 1 8 5 5 2 1.5 2 2.67 3 6 7.3 8.75 4.62 5.1 9.8 0	1.75 1.33