2015 WVU Tech High School Programming Competition

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

April 25, 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 - Wrong Answer	Your submission produced an incorrect answer for at least one of the test cases.
NO - Runtime Error	Your submission terminated abnormally.
NO - Output Format Error	The output of your program does not match the correct output.
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 charged 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 you 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

Roman Numerals

A student wants to perform simple arithmetic calculations using Roman numerals. The decimal values of the Roman numerals are:

```
M  1000
D   500
C   100
L    50
X    10
V     5
I     1
```

The student wants to find the sum, difference, product, or quotient of two Roman numerals. All operands and answers will be numbers between I and MMMCMXCIX.

Input

Each test case is described by one line of input having the format: `number operator number`. Each of these fields will be separated by a space. The operators will be one of the four basic operators: `+`, `-`, `*`, and `/`. A test case of `0` means there are no more test cases.

Output

For each case, display the case number followed by the Roman numeral result.

Sample Input	Sample Output
XCIII - XLVI CXLVIII + MMMI 0	Case 1: XLVII Case 2: MMMCXLIX

Problem B

Slope of a Tangent

You are asked to find the slope of the tangent line of a curve at a particular point. You are given a point and the equation of a polynomial. Your task is to find the slope of the tangent line of the polynomial at the point x, where x is the x-coordinate.

For example, if $x = 4.5$, and the polynomial is $3.4x^3 + 5.2x^2 + 6$, the output would be 253.35.

Input

Each test case is described by one line of input consisting of five numbers. The first number of the line is the point x. The remaining four numbers are the coefficients of a polynomial in decreasing degree. In other words, the second number is the coefficient for x^3, the third number is the coefficient for x^2, the fourth number is the coefficient for x^1, and the fifth number is the coefficient for x^0. Each number in a row is separated by a space. A test case of `0 0 0 0 0` indicates the end of input.

Output

For each case, display the case number followed by the slope of the tangent at the specified point rounded to two decimal places.

Sample Input	Sample Output

`4.5 3.4 5.2 0 6`
`56.8 0 1 -5 6`
`0 0 0 0 0`
	`Case 1: 253.35`
	`Case 2: 108.60`

Problem C

Caesear's Cipher

Caesar's cipher is a cryptography technique known as a shift cipher. It is one of the easiest encryption techniques (and possibly one of the oldest). Encryption takes each letter from the original message and shifts each letter by a set amount to a different letter. Note that the letter Z wraps around to the beginning of the alphabet in terms of shifting. A blank space will not be shifted and is simply transferred as is to the output.

For example:

- A shifted by 4 is E.
- A shifted by 2 is C.
- S shifted by 3 is V.
- Z shifted by 1 is A.

Your are asked to write a program that will convert any message by any shift value.

Input

Each line entered will contain an integer shift value (between 1 and 26), followed by a semicolon (`';'`), followed by a message that needs to be converted. The message will contain only uppercase letters. If the message contains any lowercase letters, then the program should terminate.

Output

The output should display the encrypted message on a single line with blank spaces replicated from the original input.

Sample Input	Sample Output

`4;MAY THE FORCE BE WITH YOU`
`1;ZEBRAS ARE FUN`
`exit`
	`QEC XLI JSVGI FI AMXL CSY`
	`AFCSBT BSF GTO`

Problem D

Modular Equation

You are asked to write a program that computes $(a^n) \mod d$, where a, n, and d are very large positive integers. The following theorem will be beneficial:

Theorem. *Let a, b, and d be positive integers and let $p = a \mod d$ and $q = b \mod d$. Then,*

$$(ab) \mod d = (pq) \mod d.$$

Input

Each line of input contains three numbers. The first number represents the value of a ($1 \leq a \leq 5000000000000$). The second number represents the value of n ($1 \leq n \leq 10000000$). The third number represents the value of d ($1 \leq d \leq 100000000$). An input of `0 0 0` indicates the end of the input.

Output

Output consists of a single value which is the result of $(a^n) \mod d$.

Sample Input	Sample Output
5 2 3 1232345656901 56483 23542289 2378902783120 1233879 100000000 1111111111111 1111111 11111111 0 0 0	1 10113541 0 6474909

Problem E

Sorting

Sorting a list of items is commonly needed in various programs. There are many defined algorithms for sorting a list. You need to create a sorting algorithm for words. The final list should be in reverse alphabetical order, and any duplicates from the input should be removed. The words provided can be uppercase, lowercase, or mixed case (i.e., both upper and lower). For example, the following words are considered to be equivalent (i.e., duplicates): APPLE, apple, Apple, aPple, ApPle, aPPle, ApplE.

Input

Each line will represent a single list. Each word is separated by a semicolon (`';'`). The very first thing in the list is the size of the list, and it is separated from the rest of the list with a semicolon (`';'`). If the line contains a `99`, the program should exit.

Output

The output should display the correctly ordered list in reverse alphabetical order and all duplicates should be removed. Each word should be separated with a single comma (`','`) and a space (i.e., `' '`). All capitalization should match the original input.

Sample Input

```
6;Apple;banana;Carrot;grapes;kiwi;Lettuce
7;ZEBRAS;horse;Platypus;pig;Dog;CAT;pigglet
99
```

Sample Output

```
Lettuce, kiwi, graphs, Carrot, banana, Apple
ZEBRAS, Platypus, piglet, pig, horse, Dog, CAT
```

Problem F

Palindrome Checker

A palindrome is a sequence of characters that is symmetrical. In other words, it is a string that is read identically from left to right as well as from right to left. For example, the number 12321 is a palindrome, while the number 1234 is not a palindrome. Write a program that inputs an integer and outputs whether or not the number is a palindrome.

Input

Each input will be an integer ranging from 1 to 1000000000. An input of 0 indicates that the program should terminate.

Output

For each input, the output should be either `Yes` or `No`. `Yes` implies that the input integer is a palindrome. `No` implies that the input integer is not a palindrome.

Sample Input	Sample Output
12321 1234 0	Yes No

Problem G

Matrix Multiplication

A square matrix A is an $n \times n$ grid, where each element consists of a number. An example of a 3×3 matrix is shown below:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

We denote each element of the matrix as A_{ij}, where i is the row number, and j is the column number. Using the above example,

$$A_{21} = 4.$$

Similar to numbers in general, we can add or subtract matrices. This is very simple since we add or subtract the corresponding elements in each matrix. For example, if we want to add matrices A and B, we simply add A_{ij} and B_{ij}, for all $1 \leq i, j \leq n$.

Multiplying matrices is not as simple. Suppose we want to multiply matrices A and B and store the result in matrix C, the formula is

$$c_{ij} = \sum a_{ik} \cdot b_{kj},\ 1 \leq k \leq n.$$

For example, suppose we have the matrices below:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

If we want to get c_{11}, we would have to compute

$$a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}.$$

You are asked to write a program that multiplies two $n \times n$ matrices, where $1 \leq n \leq 10$.

Input

The first line of input is the number of rows and columns of the matrix, which we denote as n, where $1 \leq n \leq 10$. The next $2n$ lines provide the data for both matrices. The first n lines give the rows of the first matrix, and each row contains n integers. The remaining n lines give the rows of the second matrix, and each row contains n integers. Each integer ranges from 0 to 100. If $n = 0$, then the program should terminate.

Output

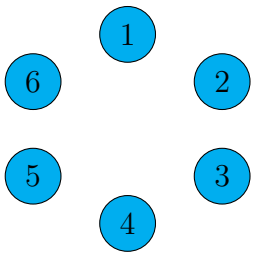
Output will consists of n rows, where each row contains n numbers. This is the product of multiplying the two input matrices.

Sample Input	Sample Output
3 1 2 3 4 5 6 7 8 9 9 8 7 6 5 4 3 2 1 2 50 75 25 30 80 90 15 35 0	30 24 18 84 69 54 138 114 90 5125 7125 2450 3300

Problem H

Ring Game

N people are playing game. They sit in a circle with N seats. The example below shows a circle with 6 players.



Starting from the person sitting on seat "1," they count off clockwise by some number M. The person who has this number loses and leaves the circle. Then, the rest continue playing the game by starting from the person who is clockwise to the person who just left. The last person remaining is the winner.

Using the above example, if $M = 3$, the first person to be eliminated is the person seated at "3", then "6", then "4", then "2", then "5." This means the person seated at "1" is the winner.

Input

Each line of input contains two integers N ($1 < N \leq 5000$) and M ($1 < M < N$). N is the number of players. M is the number to count before eliminating a player. A line with the input `0 0` indicates that the program should terminate.

Output

For each line of input, the output will be a single integer indicating the number of the winning seat.

Sample Input	Sample Output
6 3 10 6 150 25 0 0	1 3 123

Problem I

Faulty Odometer

You are given a car odometer which displays the miles traveled as an integer. However, the odometer has a defect: It proceeds from the digit 3 to the digit 5, always skipping over the digit 4. This defect shows up in all positions (the ones, the tens, the hundreds, etc.). For example, if the odometer displays 15339, and the car travels one mile, the odometer reading changes to 15350 instead of 15340.

Input

Each line of input contains a positive integer int the range of 1 to 999999999 which represents an odometer reading. Leading zeros will not appear in the input. The end of input is indicated by a line containing a single 0. You may assume that no odometer reading will contain the digit 4.

Output

Each line of input will produce exactly one line of output, which will contain: The odometer reading from the input, a colon, one blank space, and the actual number of miles traveled by the car.

Sample Input	Sample Output
13 15 2003 2005 239 250 1399 1500 999999 0	13: 12 15: 13 2003: 1461 2005: 1462 239: 197 250: 198 1399: 1052 1500: 1053 999999: 531440

Problem J

Fibonacci Sequence

The Fibonacci numbers or Fibonacci sequence are the numbers in the following integer sequence:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...$$

By definition, the first two numbers in the sequence, which we will denote as F_1 and F_2, are 1. We can compute the n^{th} Fibonacci number by computing

$$F_n = F_{n-1} + F_{n-2}.$$

For example, $F_3 = F_2 + F_1 = 1 + 1 = 2$. Further, $F_4 = F_3 + F_2 = 2 + 1 = 3$.

You are asked to write a variation of the Fibonacci sequence. Instead of letting F_1 and F_2 be 1, we can have any positive integer be F_1 and F_2. Your task is to compute the n^{th} Fibonacci number, given the first two numbers of the sequence. For example, if we set $F_1 = 3$ and $F_2 = 6$, then $F_3 = F_2 + F_1 = 3 + 6 = 9$.

Input

Each line of input will consist of three integers. The first two integers are the first two Fibonacci numbers, and each number ranges from 1 to 1000. The third number is the n^{th} Fibonacci number to compute. Note that n can range from 3 to 75.

Output

Each line of input will produce a single number for the output. The number is the n^{th} Fibonacci number where the first two numbers of the input line are the first two Fibonacci numbers.

Sample Input	Sample Output
3 6 3 3 6 4 12 16 10 0 0 0	9 15 796