Modulo Tester

Mr. Panda has been reading up about Modular Arithmetic and he finds it very interesting. Modular arithmetic has many applications such as cryptography and error detection. In modular arithmetic, calculations are performed based on a certain value called the **modulus**. Specifically, we only care about the **remainder** when the result is divided by the modulus. In Java, you can find this using the '%' operator. For example, 8 % 3 = 2 since the remainder when 8 is divided by 3 is 2. Hence, when performing modular arithmetic on some modulus **M**, the result should always be in the range **[0,M – 1]**.

Mr. Panda wants to program a class that supports the 4 modular arithemetic operations: PLUS, MINUS, TIMES, DIVIDE. After completing the class, he intends to use it for other applications such as cryptography. Addition and multiplication are straight forward in modular arithmetic, we simply perform the operation as if they were normal integers and then take the remainder when divided by the modulus. However, subtraction and division are not so straight forward.

When subtracting two numbers in modular arithmetic, if done as normal integers we might get a negative result but we want the result to be non-negative. To obtain the correct value, we need to repeatedly add the modulus **M**, until the value is in the range **[0, M-1]**. For example, if our result is -3 and the modulus is 2, we would need to add 2 twice until we get 1 which is a valid result.

Division in modular arithmetic is even more complicated. To find the value of **A ÷ B** in modulo arithmetic, we first need to find the inverse of **B** under the modulus, which we will call $\mathbf{B^{-1}}$. With that, can calculate the value of $\mathbf{A \div B = A \times B^{-1}}$. Although the inverse does not always exist making some divisions invalid (similar to dividing by zero), the divisions given in the test cases will all be valid. As calculating the inverse is difficult, Mr. Panda has helped you to write a private function that finds the inverse of a value under a modulus. If you would like to learn more about inverses, you may search the term "Modulo Inverse".

Skeleton code has provided for you to fill in your implementation of the *Modulo* class. Your job, as Mr. Panda has tasked you, is to fill in the remaining methods. In addition, your program must also include subroutines to read input and write output in the format described below.

Input
The input will contain up to **1000** lines of the same format.
The line will start with an integer representing **A**. Then a *string* denoting the operation will be given. This string will be either "PLUS", "MINUS", "TIMES" or "DIVIDE". Then, the line will end with 2 positive integers, the second integer **B** and the value of the modulus, **M**. All these will be separated with a space character.
For all our inputs, we guarantee that **A**, **B** and **M** will be non-negative and not exceed **25000**. In particular, **M** will be strictly positive.

Output
For each line, output the result of the operation in modular arithmetic. The result should be within the range **[0, M – 1]**.
For "PLUS", you are to output the value representing **A + B** (modulo **M**).
For "MINUS", you are to output the value representing **A – B** (modulo **M**)**.**
For "TIMES", you are to output the value representing **A × B** (modulo **M**).
For "DIVIDE", you are to output the value representing **A ÷ B** (modulo **M**).

Sample Input (**modulotester1.in**)	Sample Output (**modulotester1.out**)
`4 PLUS 8 5`	`2`
`7 MINUS 2 3`	`2`
`3 MINUS 8 6`	`1`
`3 TIMES 5 2`	`1`
`6 TIMES 4 8`	`0`
`2 DIVIDE 5 6`	`4`
`6 DIVIDE 4 3`	`0`

Explanation
For "2 DIVIDE 5", modulo 6: the modulo inverse of 5 is 5. 2 × 5 gives us 10. 10 gives remainder 4 when divided by 6.
For "6 DIVIDE 4", modulo 3: the modulo inverse of 4 is 1. 6 × 1 gives us 6. 6 gives remainder 0 when divided by 3.