Calculator

Rar the Cat has recently bought a scientific calculator online at a reduced price. However, he realised the calculator can only work with integers, with no support for floating point data types as well as fractions.

Being annoyed, he looked up the inner workings of his calculator and realised it is coded in Java! Rar then decides to write a "Fraction" class that supports the 4 basic arithemetic operations: PLUS, MINUS, TIMES, DIVIDE. After completing the class, he intends to add it to his calculator, so that it would support these arithmetic operations in Fractions as well.

While coding the "Fraction" class, Rar the Cat got distracted by CS2040 students and thus his code is only partially complete. As he thinks CS2040 students have exceptional programming ability, he wants you to help complete the code for the above 4 basic arithmetic functions on two fractions.

In addition, Rar the Cat does not like to compare between fractions. As such, he wants you to add 2 functions: MIN and MAX. These functions should take in 2 fractions, and output which fraction is the smaller and larger one, respectively.

Skeleton code will be provided for you to fill in your implementation of the Fraction class. In addition, your program must also include subroutines to read input and write output in the format described below.

Rar the Cat does not like non-simplified fractions where there is a common divisor between the numerator and denominator. Hence, he has helped you to write a private function to simply fractions, as well as a "toString" method for printing.

Input
The input will contain up to 100 lines of the same format.
Each line will contain 2 fractions and 1 operator. For simplicity, we will denote the first fraction as **A**, and the second fraction as **B**. Each fraction is described by 2 **positive** integers – the numerator followed by the denominator. All integers in the input will not exceed 1000.
The line will start of with 2 positive integers, the numerator and denominator of fraction **A**. Then a *string* denoting the operation/function will be given. This string will be either "PLUS", "MINUS", "TIMES", "DIVIDE", "MIN" or "MAX". Then, the line will end of with 2 positive integers, the numerator and denominator of fraction **B**. All these will be separated with a space character.

Output
For each line, output the result of the operation. As the result is a fraction, you should output the numerator, followed by the denominator, separated by a space.
For "PLUS", you are to output the Fraction representing **A + B**.
For "MINUS", you are to output the Fraction representing **A – B**.
For "TIMES", you are to output the Fraction representing **A × B**.
For "DIVIDE", you are to output the Fraction representing **A ÷ B**.
For "MIN", you are to output **A** or **B**, whichever is smaller.
For "MAX", you are to output **A** or **B**, whichever is larger.
In the provided skeleton file *Calculator.java,* the *Fraction* class has already defined "toString" to print in the required format. You are advised to print your Fraction directly. (See example in skeleton file)

Sample Input (**calculator1.in**)	Sample Output (**calculator1.out**)
2 3 TIMES 3 2 2 3 DIVIDE 3 2 2 3 PLUS 3 2 2 3 MINUS 3 2 2 3 MIN 3 2 2 3 MAX 3 2 1 1 MINUS 2 2	1 1 4 9 13 6 -5 6 2 3 3 2 0 1

Skeleton

You are given the skeleton file *Calculator.java*. You should see a non-empty file containing the partially completed class *Fraction* when opening it.

Notes:

1. You should develop your program in the subdirectory ex1 and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm.
3. You do not need to use OOP for this sit-in lab. However, you must use the *Fraction* class provided.
4. You are free to define your own helper methods. **Remember to use <u>private methods</u> whenever possible.** You are also allowed to modify the main function.
5. Please be reminded that the marking scheme is:
   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30%, which consists of:
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (incremental programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

   Compilation Error **:** Deduction of **50% of the total marks obtained**.