Rational Class

Problem:

Create a class called Rational for performing arithmetic with fractions.

Use integer variables to represent the private data of the class – the numerator and the denominator. Provide a constructor that enables an object of this class to be initialized when it's declared. The constructor should contain default values($\frac{1}{1}$) in case no initializers are provided and should store the fraction in reduced form. For example the fraction $\frac{2}{4}$ would be stored in the object as 1 in the numerator and 2 in the denominator. Provide public member functions that perform each of the following tasks:

1) add:
    a) Adding two Rational numbers. The result should be stored in reduced form.
2) sub:
    a) Subtracting two Rational numbers. The result should be stored in reduced form.
3) mul:
    a) Multiply two Rational numbers. The result should be stored in reduced form.
4) div:
    a) Dividing two Rational numbers. The result should be stored in reduced form.
5) print:
    a) Printing Rational numbers in the form a/b, where a is the numerator and b is the denominator. If a/b is a negative number, the negative sign needs to be in front, for example -3/8 is correct but 3/-8 is wrong.

Input:

Each test contains multiple test cases. The first line contains the number of test cases T ($1 \le T \le 105$). Description of the test cases follows.

Each of the following T lines contains 5 integers C, a, b, c, d. C represents the method to be done.

when C is equal to 1, you should calculate (a/b) + (c/d)
when C is equal to 2, you should calculate (a/b) - (c/d)
when C is equal to 3, you should calculate (a/b) * (c/d)
when C is equal to 4, you should calculate (a/b) / (c/d)

Output

For each test case, print in one line the number you calculated.

Sample input:

```
5
1 1 2 3 4
2 4 3 2 1
3 2 5 1 4
4 4 5 6 4
3 2 1 4 5
```

Sample output:

```
5/4
-2/3
1/10
8/15
8/5
```

Some important:

You should use the main.cpp which we gave you, and don't rewrite any code in main.cpp.

You can rewrite any code in rational.h and rational.cpp, but it must be able to successfully compile with main.cpp.

Others:

we will not use 0 as denominator, and all the numbers could be stored in int type.