C.W

Task 1:  
You are working for a scientific research team that deals extensively with ratios of measured quantities. Floating-point inaccuracies have been causing issues in the experiments, so the lead scientist requests a **fraction** (or rational number) class to represent measurements precisely.

1. **Class Requirements**:
   * Implement a Fraction class that stores the numerator and denominator as **private** integers.
   * Provide a constructor that takes (numerator, denominator).
   * Internally **reduce** the fraction to its lowest terms using the greatest common divisor (GCD).
   * Ensure the **denominator** is always positive; if the fraction is negative, store it in the numerator.
2. **Operator Overloads** (via **friend** or **member** functions):
   * Arithmetic operators:
     + operator+, operator-, operator\*, operator/.
   * Relational operators (compare two fractions):
     + operator==, operator!=, operator<, operator>, operator<=, operator>=.
   * **Insertion** and **extraction** operators to allow easy I/O:   
     friend std::ostream& operator<<(std::ostream& os, const Fraction& f);  
     friend std::istream& operator>>(std::istream& is, Fraction& f);

Example:  
**Fraction f1(2, 4), f2(3, 6);  
std::cout << (f1 + f2) << std::endl; // Should output 1 (1/2 + 1/2 = 1)**

Task 2:  
Your university’s **math department** wants a library for polynomial manipulations (addition, subtraction, multiplication, etc.). They also want certain operations (like derivative, evaluation at a point) to be handled by a **special utility class**. You will use **friend classes** so that this utility class can access polynomial internals directly.

1. **Class Requirements**:
   * A Polynomial class that stores its coefficients in a std::vector<int>—the ith element corresponds to the coefficient for x^i.
   * Constructors:
     + Default constructor (empty polynomial).
     + Constructor from a std::vector<int> of coefficients.
   * Overloads for:
     + operator+, operator-, operator\* to add, subtract, and multiply polynomials.
   * Overload operator<< to print in the form 2x^3 + 5x^2 - 1, etc.
2. **Friend Class**:
   * A class PolynomialUtils declared as a **friend** of Polynomial.
   * Methods like:
     + int evaluate(const Polynomial &p, int x);
     + Polynomial derivative(const Polynomial &p);
   * Because it’s a friend, it can read and manipulate the private coefficient array directly.
3. **Key Points**:
   * Handle negative coefficients gracefully in the operator<<.
   * Consider how to **trim** leading zeros in polynomials so you don’t print 0x^5 +

Task 3:

You’re hired by a **financial technology** firm that handles extremely large currency transactions beyond what fits in built-in integer types. They need a BigInteger class in C++ with **operator overloading** to manage sums, differences, etc.

1. **Class Requirements**:
   * BigInteger class that can store an arbitrary number of digits (use std::vector<int> to store digits from least significant to most significant, or something similar).
   * Keep the sign (positive/negative) in mind.
2. **Operator Overloads**:
   * operator+, operator-, possibly operator\* or even division for extra challenge.
   * Comparison operators: ==, !=, <, >, etc.
   * I/O operators operator<< and operator>>.
3. **Challenges**:
   * **Carry logic** in addition and multiplication.
   * Handling **negative numbers** (deciding how to store the sign).
   * Efficiently parsing input in operator>>.

H.W

Q#1

Write a program to overload decrement operator -- in such a way that when it is used

as a prefix, it multiplies a number by 4 and when it is used as a postfix then it divides

the number by 4.

Q#2

Write a program that will apply the concept of operator overloading on + operator to

add the areas of shape1 and shape2. Name of class is “shape” while shape1 and

shape2 are the objects of class shape. Use the same Area() function for both objects.

Q#3:

A company wants to calculate the bonuses of each of the employees that work in a particular department. Your services are required as a programmer to develop an automated program that will allow the company to perform their calculations. You need to create a base class called **Person** and derive two other classes named as **Admin** and **Accounts**.

* The base class will have the member functions getData, displayData and the derived class will have the member functions getData, displayData and bonus.
* The Person class will contain a data member that will store the Employee’s ID.
* The Admin and Accounts contain data members that include the name of the employee and their monthly income. The bonus function will calculate the bonus of the employees. According to the company’s policy each employee is awarded an annual bonus of 5%.
* Display each employee’s information that includes the Employee’s ID, their name, their monthly income and the bonus each one received.