Problem 3: Fraction Class

```
#include <iostream>
#include <numeric>
using namespace std;

class Fraction {
private:
    int numerator, denominator;

    void reduce() {
        int gcd_val = gcd(numerator, denominator);
        numerator /= gcd_val;
        denominator /= gcd_val;
    }

public:
    Fraction(int num = 0, int den = 1) {
        if (den == 0) throw invalid_argument("Denominator cannot be zero");
        numerator = num;
        denominator = den;
        reduce();
    }

    friend ostream& operator<<(ostream& os, const Fraction& f) {
        os << f.numerator << "/" << f.denominator;
```

```
        return os;
    }

    Fraction operator+(const Fraction& other) const {
        int num = numerator * other.denominator + other.numerator * denominator;
        int den = denominator * other.denominator;
        return Fraction(num, den);
    }

    friend Fraction operator+(int value, const Fraction& f) {
        return Fraction(value * f.denominator + f.numerator, f.denominator);
    }
};

int main() {
    cout << "test 1: ";
    Fraction f1(1, 3), f2(7, 15);
    cout << f1 << "+" << f2 << "=" << f1 + f2 << endl;

    cout << "test 2: ";
    Fraction f3(4, 5), f4(2, 5);
    cout << f3 << "+" << f4 << "=" << f3 + f4 << endl;

    cout << "test 3: ";
    int i = 1;
    Fraction f6 = i + f1;
    cout << i << "+" << f1 << "=" << f6 << endl;
```

```
    Fraction f7(2, 15), f8;

    f8 = f4 + f7;

    cout << f4 << "+" << f7 << "=" << f8 << endl;



    return 0;

}
```