```cpp
#include <iostream>
#include <string>
using namespace std;
class Rational
{
private:
    int numer;
    int denom;
public:
    int getNumer() const;
    int getDenom() const;
    void setNumer(int);
    void setDenom(int);
    void input();
    void output() const;
    Rational();
    Rational(int, int = 1);
    void reduce();
    Rational friend operator+(const Rational& a, const Rational& b);
};
void Rational::reduce()
{
    int x = abs(numer);
    int y = abs(denom);
    // find minimum of x and y
    int min = x;
    if (y < x)
        min = y;

    // finding a common factor greater than 1
    int gcf = 1;
    for (int i = 2; i <= min; i++) {
        if (x % i == 0 && y % i == 0) {
            gcf = i;
        }
    }
    numer = numer / gcf;
    denom = denom / gcf;
    if (denom < 0)
    {
        numer = -numer;
        denom = -denom;
    }
}
Rational::Rational()
{
    numer = 0;
    denom = 1;
}
```

```cpp
Rational::Rational(int x, int y)
{
    numer = x;
    if (y != 0)
        denom = y;
    else
        denom = 1;
    reduce();
}
int Rational::getNumer() const
{
    return numer;
}
int Rational::getDenom() const
{
    return denom;
}
void Rational::setNumer(int x)
{
    numer = x;
    reduce();
}
void Rational::setDenom(int x)
{
    denom = x;
    if (denom == 0)
        denom = 1;
    reduce();
}
void Rational::input()
{
    cout << "Numerator? ";
    cin >> numer;
    cout << "Denominator? ";
    cin >> denom;
    while (denom == 0)
    {
        cout << "Denominator can't be zero!\n";
        cout << "Denominator? ";
        cin >> denom;
    }
    reduce();
}
void Rational::output() const
{
    if (denom != 1)
        cout << numer << "/" << denom << endl;
    else
        cout << numer << endl;
```

```cpp
}
Rational operator+(const Rational &a, const Rational &b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getDenom() + a.getDenom() * b.getNumer());
    c.setDenom(a.getDenom() * b.getDenom());
    c.reduce();
    return c;
}
Rational operator-(const Rational& a, const Rational& b)
{
    int x = a.getNumer() * b.getDenom() - a.getDenom() * b.getNumer();
    int y= a.getDenom() * b.getDenom();
    return Rational(x,y);
}
Rational operator*(const Rational& a, const Rational& b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getNumer());
    c.setDenom(a.getDenom() * b.getDenom());
    c.reduce();
    return c;
}
Rational operator/(const Rational& a, const Rational& b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getDenom());
    c.setDenom(a.getDenom() * b.getNumer());
    c.reduce();
    return c;
}
void operator+=(Rational& a, const Rational& b)
{
    a = a + b;
}
void operator-=(Rational& a, const Rational& b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getDenom() - a.getDenom() * b.getNumer());
    c.setDenom(a.getDenom() * b.getDenom());
    c.reduce();
    a = c;
}
void operator*=(Rational& a, const Rational& b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getNumer());
    c.setDenom(a.getDenom() * b.getDenom());
    c.reduce();
    a = c;
}
void operator/=(Rational& a, const Rational& b)
{
    Rational c;
    c.setNumer(a.getNumer() * b.getDenom());
    c.setDenom(a.getDenom() * b.getNumer());
    c.reduce();
    a = c;
}
bool operator<(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) < (a.getDenom() * b.getNumer());
}
bool operator<=(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) <= (a.getDenom() * b.getNumer());
}
bool operator>(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) > (a.getDenom() * b.getNumer());
}
bool operator>=(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) >= (a.getDenom() * b.getNumer());
}
bool operator==(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) == (a.getDenom() * b.getNumer());
}
bool operator!=(const Rational& a, const Rational& b)
{
    return (a.getNumer() * b.getDenom()) != (a.getDenom() * b.getNumer());
}
Rational operator++(Rational& a) // prefix ++x
{
    a.setNumer(a.getNumer() + a.getDenom());
    return a;
}
Rational operator++(Rational& a, int n) // postfix x++
{
    Rational b = a;
    a.setNumer(a.getNumer() + a.getDenom());
    return b;
}

int main()
{
    Rational a(1, 7), b(1);
    a += b;
    a.output();
    b.output();


    return 0;
}
```