Frac(分数)类

写一个分数类，实现分数的四则运算

模拟分数的运算过程，以下是一些要注意的点

1. 构造以及运算后，需要约分，分子f和分母g需要除以它们的gcd。
2. 为了方便比较大小，将正负号表现在分子上。
3. $+=, -=, *=, /=$ 运算需要返回对象的引用。
4. 比较大小转变为乘法，比较大小。

```
#include<bits/stdc++.h>

using ll = long long;

template<class T>
class Frac {
    T f, g;
    void _to(T x, T y) {
        T gcd = std::gcd(std::abs(x), std::abs(y));
        if (y < 0) {
            y = -y;
            x = -x;
        }
        f = x / gcd, g = y / gcd;
    }

    public:

    Frac() {
        f = 0, g = 1;
    }

    Frac(T _f, T _g) {
        _to(_f, _g);
    }

    Frac(T _num) : Frac(_num, 1) {}

    Frac(const Frac &rhs) {
        f = rhs.f;
        g = rhs.g;
    }

    Frac operator-() {
        return {-f, g};
    }

    Frac &operator+=(const Frac &rhs) {
        f = f * rhs.g + rhs.f * g;
        g *= rhs.g;
        _to(f, g);
        return *this;
    }
```

```
Frac &operator-=(const Frac &rhs) {
    f = f * rhs.g - rhs.f * g;
    g *= rhs.g;
    _to(f, g);
    return *this;
}

Frac &operator*=(const Frac &rhs) {
    f *= rhs.f;
    g *= rhs.g;
    _to(f, g);
    return *this;
}

Frac &operator/=(const Frac &rhs) {
    f *= rhs.g;
    g *= rhs.f;
    _to(f, g);
    return *this;
}

friend Frac operator+(Frac lhs, const Frac &rhs) {
    return lhs += rhs;
}
friend Frac operator-(Frac lhs, const Frac &rhs) {
    return lhs -= rhs;
}
friend Frac operator*(Frac lhs, const Frac &rhs) {
    return lhs *= rhs;
}
friend Frac operator/(Frac lhs, const Frac &rhs) {
    return lhs /= rhs;
}

friend bool operator==(const Frac &lhs, const Frac &rhs) {
    return lhs.f * rhs.g == rhs.f * lhs.g;
}
friend bool operator!=(const Frac &lhs, const Frac &rhs) {
    return lhs.f * rhs.g != rhs.f * lhs.g;
}
friend bool operator<(const Frac &lhs, const Frac &rhs) {
    return lhs.f * rhs.g < rhs.f * lhs.g;
    }
    friend bool operator>(const Frac &lhs, const Frac &rhs) {
        return lhs.f * rhs.g > rhs.f * lhs.g;
    }
    friend bool operator<=(const Frac &lhs, const Frac &rhs) {
        return lhs.f * rhs.g <= rhs.f * lhs.g;
    }
    friend bool operator>=(const Frac &lhs, const Frac &rhs) {
        return lhs.f * rhs.g >= rhs.f * lhs.g;
    }

    // 这里也可以用<=>运算符
    /*friend strong_ordering operator <=>(const Frac &lhs, const Frac &rhs) {
        return lhs.f * rhs.g <=> rhs.f * lhs.g;
    }*/

    friend std::ostream &operator<<(std::ostream &out, Frac x) {
        if (x.g == 1) {
            return out << x.f;
        } else {
            return out << x.f << "/" << x.g;
        }
    }

    std::pair<T, T> val() {
        return {f, g};
    }
};

using Z = Frac<ll>;
```