```
ModularInteger operator / (ModularInteger &x) {
        ModularInteger res;
        res.val = val;
        res = res * binexp(x, MOD - 2);
        return res;
}

void operator += (ModularInteger const &x) {
        val = val + x.val;
        if (val >= MOD) val -= MOD;
}

void operator -= (ModularInteger const &x) {
        val = val - x.val;
        if (val < 0) val += MOD;
}

void operator *= (ModularInteger const &x) {
        val = val * x.val;
        val %= MOD;
}

void operator /= (ModularInteger &x) {
        ModularInteger res;
        res.val = val;
        res = res * binexp(x, MOD - 2);
        val = res.val;
}

void operator = (long long x) {
        x %= MOD;
        if (x < 0) x += MOD;
        val = x;
}

ModularInteger operator + (long long x) {
        x %= MOD;
        if (x < 0) x += MOD;
        ModularInteger res;
        res.val = val + x;
        if (res.val >= MOD) res.val -= MOD;
        return res;
}
```