```
/*
    Due: October 1, 2018

    Assignment: Implement the following class. Make sure that it handles all signed 8-byte
integers correctly and without overflow.

    For now, if the user attempts to divide by zero or set the modulus to something < 2, you
can just "fail fast" with exit(-1).
 */

#ifndef ____Mod__
#define ____Mod__

#include <iostream>
#include <cassert>

using std::istream;
using std::ostream;
using std::cin;
using std::cout;
using std::cerr;

class Mod {
public:
    Mod(long t);
    Mod(const Mod& m);
    Mod& operator=(const Mod& m);
    Mod& operator+=(const Mod& m);
    Mod& operator-=(const Mod& m);
    Mod& operator*=(const Mod& m);
    Mod& operator/=(const Mod& m);
    Mod operator-() const;
    Mod pwr(long e) const;
    long val() const;

    static void set_modulus(long m);
    static long get_modulus() { return modulus; }

private:
    long x;
    static long modulus;

    static Mod inv(long r0);
};


Mod operator+(const Mod& a, const Mod& b);
Mod operator+(long t, const Mod& m);
Mod operator-(const Mod& a, const Mod& b);
Mod operator-(long t, const Mod& m);
Mod operator*(const Mod& a, const Mod& b);
Mod operator*(long t, const Mod& m);
Mod operator/(const Mod& a, const Mod& b);
Mod operator/(long t, const Mod& m);
bool operator==(const Mod& a, const Mod& b);
bool operator==(long t, const Mod& m);
bool operator!=(const Mod& a, const Mod& b);
bool operator!=(long t, const Mod& m);

istream& operator>>(istream& is, Mod& m);
ostream& operator<<(ostream& os, const Mod& m);

//long Mod::modulus = 17;

#endif /* defined(____Mod__) */
```