```
//      Author       :    Jan
//      Problem Name :    Big int for contest

#include <string>
#include <algorithm>
                                                                        string print() {
using namespace std;                                                            string res = a;
                                                                                if (signo == -1) res.push_back('-');
class grande {                                                                  reverse(res.begin(), res.end());
public:                                                                         return res;
        string a; //atributo                                            }
        int signo; //atributo
        grande(){}
        grande(string b){ (*this) = b; }
        int size(){ return a.size(); }
        grande invertirSigno(){ signo *= -1; return (*this); }
        grande normalizar(int nuevoSigno){
                signo = nuevoSigno;
                for (int i = a.size() - 1; i > 0 && a[i] == '0'; --i)
                        a.erase(a.begin() + i);
                if (a.size() == 1 && a[0] == '0' ) signo = 1;
                return (*this);
        }
        void operator = (string b){
                a = b[0] == '-' ? b.substr(1) : b;
                reverse(a.begin(), a.end());
                this->normalizar(b[0] == '-' ? -1 : 1);
        }
        bool operator < (const grande &b) const {
                if (a.size() != b.a.size()) return a.size() < b.a.size();
                for (int i = a.size() - 1; i >= 0; --i)
                        if (a[i] != b.a[i]) return a[i] < b.a[i];
                return false;
        }
        grande operator + (grande b){
                if (signo != b.signo) return (*this) - b.invertirSigno();
                grande c;
                for (int i = 0, carry = 0; i < (int)a.size() || i < (int)b.size() || carry; ++i){
                        carry += (i < (int)a.size() ? a[i] - 48 : 0) + (i < (int)b.a.size() ? b.a[i] - 48 : 0);
                        c.a += (carry % 10 + 48);
                        carry /= 10;
                }
                return c.normalizar(signo);
        }
        grande operator - (grande b){
                if (signo != b.signo) return (*this) + b.invertirSigno();
                if ((*this) < b) return (b - (*this)).invertirSigno();
                grande c;
                for (int i = 0, borrow = 0; i < (int)a.size(); ++i){
                        borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
                        c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
                        borrow = borrow >= 0 ? 0 : 1;
                }
                return c.normalizar(signo);
        }
        grande operator * (grande b){
                grande c("0");
                for (int i = 0, k = a[i]; i < (int)a.size(); ++i, k = a[i]){
                        while (k-- - 48) c = c + b;
                        b.a.insert(b.a.begin(), '0');
                }
                return c.normalizar(signo * b.signo);
        }
        grande operator / (grande b){
                if (b.size() == 1 && b.a[0] == '0') b.a[0] /= (b.a[0] - 48);
                grande c("0"), d;
                for (int j = 0; j < (int)a.size(); ++j) d.a += "0";
                int dSign = signo * b.signo; b.signo = 1;
                for (int i = a.size() - 1; i >= 0; --i){
                        c.a.insert(c.a.begin(), '0');
                        c = c + a.substr(i, 1);
                        while (!( c < b )) c = c - b, d.a[i]++;
                }
                return d.normalizar(dSign);
        }
        grande operator % (grande b){
                if (b.size() == 1 && b.a[0] == '0') b.a[0] /= (b.a[0] - 48);
                grande c("0");
                int cSign = signo * b.signo; b.signo = 1;
                for (int i = a.size() - 1; i >= 0; --i){
                        c.a.insert(c.a.begin(), '0');
                        c = c + a.substr(i, 1);
                        while (!( c < b )) c = c - b;
                }
                return c.normalizar(cSign);
        }
};
```