高精度

Big Interger

```cpp
#include<algorithm>
#include<iostream>
#include<cstring>
#include<vector>
#include<cstdio>

using namespace std;

typedef long long LL;
#define pb push_back

struct BigNum{
    vector<int> a; // a[n-1]...a[1]a[0]
    int neg;

    BigNum(){ a.clear(); neg = 1; }
    explicit BigNum(const string &s){
        a.clear();
        int len = s.length();
        for(int i = 0; i < len; i++)
            a.pb(s[len-i-1] - '0');
    }
    explicit BigNum(LL num){
        a.clear();
        do{
            a.pb(num % 10);
            num /= 10;
        }while(num);
    }

    BigNum operator = (const string &s){ return *this = BigNum(s); }
    BigNum operator = (LL num){ return *this = BigNum(num); }

    bool operator < (const BigNum &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;
    }
    bool operator > (const BigNum &b) const{ return b < *this; }
    bool operator <= (const BigNum &b) const{ return !(*this > b); }
    bool operator >= (const BigNum &b) const{ return !(*this < b); }
    bool operator != (const BigNum &b) const{ return (*this > b) || (*this < b); }
    bool operator == (const BigNum &b) const{ return !(*this < b) && !(*this > b); }

    BigNum operator + (const BigNum &b) const{
        BigNum C;
        int x = 0;
        for(int i = 0, g = 0; ; i++){
            if(g == 0 && i >= a.size() && i >= b.a.size())  break;
            x = g;
            if(i < a.size())    x += a[i];
            if(i < b.a.size())  x += b.a[i];
            C.a.pb(x % 10);
            g = x / 10;
        }
        return C;
    }
    BigNum operator - (const BigNum &b) const{
        BigNum C;
        BigNum A = *this;
        BigNum B = b;
        if(A < B)   C.neg = -1, swap(A, B);
        C.a.resize(A.a.size());
        for(int i = 0; ; i++){
            if(i >= A.a.size() && i >= B.a.size())  break;
            if(i >= B.a.size()) C.a[i] = A.a[i];
            else    C.a[i] = A.a[i] - B.a[i];
        }
        for(int i = 0; ; i++){
            if(i >= C.a.size()) break;
            if(C.a[i] < 0){
                C.a[i] += 10;
                C.a[i+1]--;
            }
        }
        while(C.a.size() > 1 && C.a.back() == 0)    C.a.pop_back();
```

```
            return C;
        }
        BigNum operator * (const BigNum &b) const{
            BigNum C;
            C.a.resize(a.size() + b.a.size());
            for(int i = 0; i < a.size(); i++){
                int g = 0;
                for(int j = 0; j < b.a.size(); j++){
                    C.a[i+j] += a[i] * b.a[j] + g;
                    g = C.a[i+j] / 10;
                    C.a[i+j] %= 10;
                }
                C.a[i+b.a.size()] = g;
            }
            while(C.a.size() > 1 && C.a.back() == 0)    C.a.pop_back();
            return C;
        }
        BigNum operator / (const LL &b) const{
            BigNum C;
            C = *this;
            for(int i = C.a.size() - 1; i >= 0; i--){
                if(i)   C.a[i-1] += C.a[i] % b * 10;
                C.a[i] /= b;
            }
            while(C.a.size() > 1 && C.a.back() == 0)
                C.a.pop_back();
            return C;
        }
        BigNum operator / (const BigNum &b) const{
            BigNum L, R, ans, t;
            L = 0ll;
            R = *this;
            ans = 0ll;
            t = 1ll;
            while(L <= R){
                BigNum mid = (L + R) / 2;
                if((mid * b) > (*this))
                    R = mid - t;
                else
                    L = mid + t, ans = mid;
            }
            return ans;
        }
        BigNum operator % (const LL &b) const{
            BigNum B; B = b;
            return (*this) - (*this) / b * B;
        }
```

```cpp
    BigNum operator % (const BigNum &b) const{
        return (*this) - (*this) / b * b;
    }
    BigNum operator += (const BigNum &b){ *this = *this + b; return *this; }
    BigNum operator -= (const BigNum &b){ *this = *this - b; return *this; }
    BigNum operator *= (const BigNum &b){ *this = *this * b; return *this; }
    BigNum operator /= (const LL &b){ *this = *this / b; return *this; }
    BigNum operator /= (const BigNum &b){ *this = *this / b; return *this; }

};

ostream& operator << (ostream &out, const BigNum &b){
    string res;
    if(b.neg == -1) res += '-';
    for(int i = b.a.size() - 1; i >= 0; i--)
        res += b.a[i] + '0';
    return out << res;
}
istream& operator >> (istream &in, BigNum &b){
    string str;
    if(in >> str)   b = str;
    return in;
}

BigNum s1, s2;

int main(){
    BigNum a, b;
    cin >> a >> b;
    cout << a + b << endl;
    cout << a - b << endl;
    cout << a * b << endl;
    cout << a / b << endl;
    cout << a % b << endl;
    return 0;
}
```