

高精度

Big Integer

```
1  #include<algorithm>
2  #include<iostream>
3  #include<cstring>
4  #include<vector>
5  #include<cstdio>
6
7  using namespace std;
8
9  typedef long long LL;
10 #define pb push_back
11
12 struct BigNum{
13     vector<int> a; // a[n-1]...a[1]a[0]
14     int neg;
15
16     BigNum(){ a.clear(); neg = 1; }
17     explicit BigNum(const string &s){
18         a.clear();
19         int len = s.length();
20         for(int i = 0; i < len; i++){
21             a.pb(s[len-i-1] - '0');
22         }
23     }
24     explicit BigNum(LL num){
25         a.clear();
26         do{
27             a.pb(num % 10);
28             num /= 10;
29         }while(num);
30     }
31
32     BigNum operator = (const string &s){ return *this = BigNum(s); }
33     BigNum operator = (LL num){ return *this = BigNum(num); }
34
35     bool operator < (const BigNum &b) const{
36         if(a.size() != b.a.size()) return a.size() < b.a.size();
37         for(int i = a.size() - 1; i >= 0; i--){
38             if(a[i] != b.a[i])
39                 return a[i] < b.a[i];
40         }
41         return false;
42     }
43
44     bool operator > (const BigNum &b) const{ return b < *this; }
45     bool operator <= (const BigNum &b) const{ return !(*this > b); }
46     bool operator >= (const BigNum &b) const{ return !(*this < b); }
47     bool operator != (const BigNum &b) const{ return (*this > b) || (*this < b); }
48     bool operator == (const BigNum &b) const{ return !(*this < b) && !(*this > b); }
49
50     BigNum operator + (const BigNum &b) const{
51         BigNum C;
52         int x = 0;
53         for(int i = 0, g = 0; ; i++){
54             if(g == 0 && i >= a.size() && i >= b.a.size()) break;
55             x = g;
56             if(i < a.size()) x += a[i];
57             if(i < b.a.size()) x += b.a[i];
58             C.a.pb(x % 10);
59             g = x / 10;
```

```

57     }
58     return C;
59 }
60 BigNum operator - (const BigNum &b) const{
61     BigNum C;
62     BigNum A = *this;
63     BigNum B = b;
64     if(A < B)    C.neg = -1, swap(A, B);
65     C.a.resize(A.a.size());
66     for(int i = 0; ; i++){
67         if(i >= A.a.size() && i >= B.a.size()) break;
68         if(i >= B.a.size()) C.a[i] = A.a[i];
69         else    C.a[i] = A.a[i] - B.a[i];
70     }
71     for(int i = 0; ; i++){
72         if(i >= C.a.size()) break;
73         if(C.a[i] < 0){
74             C.a[i] += 10;
75             C.a[i+1]--;
76         }
77     }
78     while(C.a.size() > 1 && C.a.back() == 0)    C.a.pop_back();
79     return C;
80 }
81 BigNum operator * (const BigNum &b) const{
82     BigNum C;
83     C.a.resize(a.size() + b.a.size());
84     for(int i = 0; i < a.size(); i++){
85         int g = 0;
86         for(int j = 0; j < b.a.size(); j++){
87             C.a[i+j] += a[i] * b.a[j] + g;
88             g = C.a[i+j] / 10;
89             C.a[i+j] %= 10;
90         }
91         C.a[i+b.a.size()] = g;
92     }
93     while(C.a.size() > 1 && C.a.back() == 0)    C.a.pop_back();
94     return C;
95 }
96 BigNum operator / (const LL &b) const{
97     BigNum C;
98     C = *this;
99     for(int i = C.a.size() - 1; i >= 0; i--){
100         if(i)    C.a[i-1] += C.a[i] % b * 10;
101         C.a[i] /= b;
102     }
103     while(C.a.size() > 1 && C.a.back() == 0)
104         C.a.pop_back();
105     return C;
106 }
107 BigNum operator / (const BigNum &b) const{
108     BigNum L, R, ans, t;
109     L = 0ll;
110     R = *this;
111     ans = 0ll;
112     t = 1ll;
113     while(L <= R){
114         BigNum mid = (L + R) / 2;
115         if((mid * b) > (*this))
116             R = mid - t;
117         else
118             L = mid + t, ans = mid;
119     }
120     return ans;
121 }
122 BigNum operator % (const LL &b) const{
123     BigNum B; B = b;

```

```

124         return (*this) - (*this) / b * B;
125     }
126     BigNum operator % (const BigNum &b) const{
127         return (*this) - (*this) / b * b;
128     }
129     BigNum operator += (const BigNum &b){ *this = *this + b; return *this; }
130     BigNum operator -= (const BigNum &b){ *this = *this - b; return *this; }
131     BigNum operator *= (const BigNum &b){ *this = *this * b; return *this; }
132     BigNum operator /= (const LL &b){ *this = *this / b; return *this; }
133     BigNum operator /= (const BigNum &b){ *this = *this / b; return *this; }
134
135 };
136
137 ostream& operator << (ostream &out, const BigNum &b){
138     string res;
139     if(b.neg == -1) res += '-';
140     for(int i = b.a.size() - 1; i >= 0; i--){
141         res += b.a[i] + '0';
142     }
143     return out << res;
144 }
145 istream& operator >> (istream &in, BigNum &b){
146     string str;
147     if(in >> str)    b = str;
148     return in;
149 }
150
151 BigNum s1, s2;
152
153 int main(){
154     BigNum a, b;
155     cin >> a >> b;
156     cout << a + b << endl;
157     cout << a - b << endl;
158     cout << a * b << endl;
159     cout << a / b << endl;
160     cout << a % b << endl;
161     return 0;
162 }

```