**CPP\_BigInt**

struct Bigint

{

string a;

int sign;

Bigint() {}

Bigint( string b ) { (\*this) = b; }

int size() { return a.size(); }

Bigint inverseSign() { sign \*= -1; return (\*this);}

Bigint normalize( int newSign )

{

for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )

a.erase(a.begin() + i);

sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;

return (\*this);

}

void operator = ( string b )

{

a = b[0] == '-' ? b.substr(1) : b;

reverse( a.begin(), a.end() );

this->normalize( b[0] == '-' ? -1 : 1 );

}

bool operator < ( const Bigint &b ) const {

if( sign != b.sign ) return sign < b.sign;

if( a.size() != b.a.size() )

return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();

for( int i = a.size() - 1; i >= 0; i-- ) if( a[i] != b.a[i] )

return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];

return false;

}

bool operator == ( const Bigint &b ) const {

return a == b.a && sign == b.sign;

}

Bigint operator + ( Bigint b ) {

if( sign != b.sign ) return (\*this) - b.inverseSign();

Bigint c;

for(int i = 0, carry = 0; i<a.size() || i<b.size() || carry; i++ )

{

carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size() ? b.a[i]-48 : 0);

c.a += (carry % 10 + 48);

carry /= 10;

}

return c.normalize(sign);

}

Bigint operator - ( Bigint b ) {

if( sign != b.sign ) return (\*this) + b.inverseSign();

int s = sign; sign = b.sign = 1;

if( (\*this) < b ) return ((b - (\*this)).inverseSign()).normalize(-s);

Bigint c;

for( int i = 0, borrow = 0; i < 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.normalize(s);

}

Bigint operator \* ( Bigint b ) {

Bigint c("0");

for( int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48 )

{

while(k--) c = c + b;

b.a.insert(b.a.begin(), '0');

}

return c.normalize(sign \* b.sign);

}

Bigint operator / ( Bigint b ) {

if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= ( b.a[0] - 48 );

Bigint c("0"), d;

for( int j = 0; j < a.size(); j++ ) d.a += "0";

int dSign = sign \* b.sign; b.sign = 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.normalize(dSign);

}

Bigint operator % ( Bigint b ) {

if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= ( b.a[0] - 48 );

Bigint c("0");

b.sign = 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.normalize(sign);

}

void print()

{

if( sign == -1 ) putchar('-');

for( int i = a.size() - 1; i >= 0; i-- ) putchar(a[i]);

}

};