Big Integer

const int base = 1e9;

typedef vector<int> BigInt;

Reset

void Set(BigInt &a){

while(a.size() > 1 && a.back() == 0)

a.pop\_back();

}

Print

void Print(BigInt a){

Set(a);

if(!a.size()) puts("0");

else{

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

printf("%d", a[i]);

puts("");

}

}

Constructor String

BigInt Integer(string s){

BigInt ans;

if(s[0] == '-') return ans;

if(s.size() == 0) {ans.push\_back(0); return ans;}

while(s.size()%9 != 0) s = '0' + s;

for(int i=0; i<s.size(); i+=9){

int v = 0;

for(int j=i; j<i+9; j++) v = v\*10 + (s[j] - '0');

ans.insert(ans.begin(), v);

}

Set(ans);

return ans;

}

Constructor Long Long

BigInt Integer(long long x){

string s = "";

while(x) s = char(x%10 + '0') + s, x /= 10;

return Integer(s);

}

Constructor Integer

BigInt Integer(int x){

return Integer((long long)x);

}

Input

void operator >> (istream &in, BigInt &a){

string s; in>>s;

a = Integer(s);

}

Output

void operator << (ostream &out, BigInt a){Print(a);}

Comparing Operators

bool operator < (BigInt a, BigInt b){

Set(a); Set(b);

if(a.size() != b.size()) return a.size() < b.size();

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

if(a[i] != b[i])

return a[i] < b[i];

return false;

}

bool operator > (BigInt a, BigInt b){return (b < a);}

bool operator == (BigInt a, BigInt b){return !(a<b) && !(b < a);}

bool operator <= (BigInt a, BigInt b){return a<b || a==b;}

bool operator >= (BigInt a, BigInt b){return b<a || a==b;}

bool operator < (BigInt a, int b){return a<Integer(b);}

bool operator > (BigInt a, int b){return a>Integer(b);}

bool operator == (BigInt a, int b){return a == Integer(b);}

bool operator >= (BigInt a, int b){return a >= Integer(b);}

bool operator <= (BigInt a, int b){return a <= Integer(b);}

bool operator < (BigInt a, long long b){return a<Integer(b);}

bool operator > (BigInt a, long long b){return a>Integer(b);}

bool operator == (BigInt a, long long b){return a == Integer(b);}

bool operator >= (BigInt a, long long b){return a >= Integer(b);}

bool operator <= (BigInt a, long long b){return a <= Integer(b);}

bool operator < (BigInt a, string b){return a<Integer(b);}

bool operator > (BigInt a, string b){return a>Integer(b);}

bool operator == (BigInt a, string b){return a == Integer(b);}

bool operator >= (BigInt a, string b){return a >= Integer(b);}

bool operator <= (BigInt a, string b){return a <= Integer(b);}

Max / Min

BigInt max(BigInt a, BigInt b){return a>b?a:b;}

BigInt min(BigInt a, BigInt b){return a<b?a:b;}

Arithmetic Operators

Plus

BigInt operator + (BigInt a, BigInt b){

Set(a); Set(b);

BigInt ans; int carry = 0, sz = max(a.size(), b.size());

for(int i=0; i<sz; i++){

if(i < a.size()) carry += a[i];

if(i < b.size()) carry += b[i];

ans.push\_back(carry%base);

carry /= base;

}

if(carry) ans.push\_back(carry);

Set(ans);

return ans;

}

BigInt operator + (BigInt a, int b) {return a+Integer(b);}

BigInt operator + (BigInt a, long long b) {return a+Integer(b);}

BigInt operator + (BigInt a, string b) {return a+Integer(b);}

BigInt operator ++ (BigInt &a){return a = a + 1;}

BigInt operator += (BigInt &a, BigInt b) {return a = a + b;}

BigInt operator += (BigInt &a, int b) {return a = a + Integer(b);}

BigInt operator += (BigInt &a, string b) {return a = a + Integer(b);}

BigInt operator += (BigInt &a, long long b) {return a = a + Integer(b);}

Minus

BigInt operator - (BigInt a, BigInt b){

Set(a); Set(b);

BigInt ans; int carry = 0;

for(int i=0; i<a.size(); i++){

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

if(carry < 0) ans.push\_back(carry + base), carry = -1;

else ans.push\_back(carry), carry = 0;

}

Set(ans); return ans;

}

BigInt operator - (BigInt a, int b) {return a-Integer(b);}

BigInt operator - (BigInt a, long long b) {return a-Integer(b);}

BigInt operator - (BigInt a, string b) {return a-Integer(b);}

BigInt operator -- (BigInt &a){return a = a - 1;}

BigInt operator -= (BigInt &a, BigInt b) {return a = a - b;}

BigInt operator -= (BigInt &a, int b) {return a = a - Integer(b);}

BigInt operator -= (BigInt &a, string b) {return a = a - Integer(b);}

BigInt operator -= (BigInt &a, long long b) {return a = a - Integer(b);}

Multiplication

BigInt operator \* (BigInt a, BigInt b){

Set(a); Set(b);

BigInt ans;

ans.assign(a.size() + b.size(), 0);

for(int i=0; i<a.size(); i++){

long long carry = 0LL;

for(int j=0; j<b.size() || carry > 0; j++){

long long s = ans[i+j] + carry + (long long)a[i] \* (j < b.size() ? (long long)b[j]:0LL);

ans[i+j] = s % base;

carry = s / base;

}

}

Set(ans); return ans;

}

BigInt operator \* (BigInt a, int b){return a \* Integer(b);}

BigInt operator \* (BigInt a, string b){return a \* Integer(b);}

BigInt operator \* (BigInt a, long long b){return a \* Integer(b);}

BigInt operator \*= (BigInt &a, int b){return a = a \* Integer(b);}

BigInt operator \*= (BigInt &a, string b){return a = a \* Integer(b);}

BigInt operator \*= (BigInt &a, long long b){return a = a \* Integer(b);}

Divison

BigInt operator / (BigInt a, BigInt b){

Set(a); Set(b);

if(b == Integer(0)) return Integer(0);

BigInt ans, cur;

for(int i=a.size()-1; i>=0; i--){

cur.insert(cur.begin(), a[i]);

int x = 0, L = 0, R = base;

while(L <= R){

int mid = (L+R) >> 1;

if(b \* Integer(mid) > cur){

x = mid;

R = mid - 1;

}

else L = mid + 1;

}

cur = cur - Integer(x-1) \* b;

ans.insert(ans.begin(), x-1);

}

Set(ans); return ans;

}

BigInt operator / (BigInt a, string b){return a / Integer(b);}

BigInt operator / (BigInt a, long long b){

Set(a); BigInt ans;

long long cur = 0LL;

for(int i=a.size()-1; i>=0; i--){

cur = (cur \* (long long)base + (long long)a[i]);

ans.insert(ans.begin(), cur/b);

cur %= b;

}

Set(ans); return ans;

}

BigInt operator / (BigInt a, int b){return a / (long long)b;}

BigInt operator /= (BigInt &a, BigInt b){return a = a/b;}

BigInt operator /= (BigInt &a, int b){return a = a/b;}

BigInt operator /= (BigInt &a, string b){return a = a/b;}

BigInt operator /= (BigInt &a, long long b){return a = a/b;}

Moduler

BigInt operator % (BigInt a, BigInt b){

Set(a); Set(b);

if(b == Integer(0)) return Integer(-1);

BigInt ans;

for(int i=a.size()-1; i>=0; i--){

ans.insert(ans.begin(), a[i]);

int x = 0, L = 0, R = base;

while(L <= R){

int mid = (L+R) >> 1;

if(b \* Integer(mid) > ans){

x = mid;

R = mid - 1;

}

else L = mid + 1;

}

ans = ans - Integer(x-1) \* b;

}

Set(ans); return ans;

}

BigInt operator % (BigInt a, string b) {return a%Integer(b);}

long long operator % (BigInt a, long long b){

Set(a);

if(!b) return -1;

int ans = 0;

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

ans = (ans \* (base%b) + a[i]%b)%b;

return ans;

}

long long operator % (BigInt a, int b){return a%(long long)b;}

BigInt operator %= (BigInt &a, BigInt b){return a = a%b;}

BigInt operator %= (BigInt &a, string b){return a = a%b;}

BigInt operator %= (BigInt &a, int b){return a = a%Integer(b);}

BigInt operator %= (BigInt &a, long long b){return a = a%Integer(b);}

GCD

BigInt gcd(BigInt a, BigInt b){

Set(a); Set(b);

while(b > Integer(0)){

BigInt r = a%b;

a = b;

b = r;

}

Set(a); return a;

}

LCM

BigInt lcm(BigInt a, BigInt b){return (a\*b / gcd(a,b));}

SQRT

BigInt sqrt(BigInt a){

BigInt x0 = a, x1 = (a+1)/2;

while(x1 < x0){

x0 = x1;

x1 = (x1 + a/x1)/2;

}

return x0;

}

POWER( log n)

BigInt power(BigInt a, BigInt b){

if(b == Integer(0)) return Integer(1);

BigInt tmp = power(a, b/2);

if(b%2 == 0) return tmp \* tmp;

return tmp \* tmp \* a;

}

BigInt power(BigInt a, int b) {return power(a, Integer(b));}

BigInt power(BigInt a, string b) {return power(a, Integer(b));}

BigInt power(BigInt a, long long b) {return power(a, Integer(b));}

Logarithm

int log(int n, BigInt a){

Set(a); int ans = 0;

while(a > Integer(1)){

ans++;

a /= n;

}

return ans;

}