// 高精度

const int BIT=1000;

inline int getw(int x)

{

if (!x) return 1;

int res=0;

while (x) res++,x/=10;

return res;

}

inline string INT\_TO\_STRING(ull x)

{

string res;res="";

if (!x) {res="0";return res;}

while (x)

{

res=string(1,'0'+x%10)+res;

x/=10;

}

return res;

}

struct Bigint

{

int b[108],len;

Bigint () {memset(b,0,sizeof(b));}

inline bool iszero() {return len==1 && b[1]==0;}

inline void Set(int x)

{

memset(b,0,sizeof(b));

if (!x) {b[1]=0;len=1;return;}

len=0;

while (x) b[++len]=x%BIT,x/=BIT;

}

inline string trans()

{

string res;res="";

for (register int i=len;i>=1;i--)

{

int nu=getw(b[i]);string tmp=INT\_TO\_STRING(b[i]);

if (i!=len) for (register int j=1;j<=(3-nu);j++) tmp="0"+tmp;

res+=tmp;

}

return res;

}

inline bool operator < (Bigint other)

{

if (len<other.len) return true;

if (len>other.len) return false;

for (register int i=len;i>=1;i--)

{

if (b[i]<other.b[i]) return true;

if (b[i]>other.b[i]) return false;

}

return false;

}

inline bool operator == (Bigint other)

{

if (len!=other.len) return false;

for (register int i=1;i<=len;i++) if (b[i]!=other.b[i]) return false;

return true;

}

inline bool operator > (Bigint other)

{

if (len<other.len) return false;

if (len>other.len) return true;

for (register int i=len;i>=1;i--)

{

if (b[i]>other.b[i]) return true;

if (b[i]<other.b[i]) return false;

}

return false;

}

inline bool operator <= (Bigint other) {return \*(this)<other || \*(this)==other;}

inline bool operator >= (Bigint other) {return \*(this)>other || \*(this)==other;}

inline Bigint operator << (int num)

{

Bigint res;res=\*(this);

if (len==1 && !b[1]) return res;

for (register int i=len;i>=1;i--) res.b[i+num]=res.b[i];

res.len+=num;return res;

}

inline Bigint operator + (Bigint other)

{

Bigint res;res.Set(0);

res.len=max(len,other.len);

for (register int i=1;i<=res.len;i++)

{

res.b[i]+=b[i]+other.b[i];

res.b[i+1]+=res.b[i]/BIT;

res.b[i]%=BIT;

}

if (res.b[res.len+1]) res.len++;

return res;

}

inline Bigint operator - (Bigint other)

{

assert(len>=other.len);

Bigint res;res.Set(0);

for (register int i=1;i<=len;i++)

{

if (b[i]<other.b[i]) b[i]+=BIT,b[i+1]--;

res.b[i]=b[i]-other.b[i];

}

res.len=len;

while (res.len && res.b[res.len]==0) res.len--;

if (!res.len) res.len=1;

return res;

}

inline Bigint operator \* (Bigint other)

{

Bigint res;res.Set(0);

if ((len==1 && b[len]==0) || other.iszero()) return res;

for (register int i=1;i<=len;i++)

for (register int j=1;j<=other.len;j++)

res.b[i+j-1]+=b[i]\*other.b[j];

res.len=len+other.len-1;

for (register int i=1;i<=res.len;i++) res.b[i+1]+=res.b[i]/BIT,res.b[i]%=BIT;

if (res.b[res.len+1]) res.len++;

return res;

}

inline int getbit(Bigint other)

{

Bigint tmp;int l=0,r=BIT-1,mid,ans;

while (l<=r)

{

mid=(l+r)>>1;

tmp.Set(mid);if (other\*tmp<=\*(this)) ans=mid,l=mid+1; else r=mid-1;

}

return ans;

}

inline Bigint operator / (Bigint other)

{

Bigint cur;cur.Set(0);Bigint res,tmp;

res.len=len;

for (register int i=len;i>=1;i--)

{

cur=(cur<<1);cur.b[1]=b[i];

int w=cur.getbit(other);

res.b[i]=w;tmp.Set(w);

cur=cur-other\*tmp;

}

while (!res.b[res.len]) res.len--;

return res;

}

inline Bigint operator % (Bigint other)

{

Bigint cur;cur.Set(0);Bigint tmp;

for (register int i=len;i>=1;i--)

{

cur=(cur<<1);cur.b[1]=b[i];

int w=cur.getbit(other);

tmp.Set(w);

cur=cur-other\*tmp;

}

while (cur.len && !cur.b[cur.len]) cur.len--;

if (!cur.len) cur.len++;

return cur;

}

};