#include<cstdio>

#include <cstring>

/\*

Title：整数高精度模板

Arranged by：bin\_gege

functions: 完美支持正负数的高精度运算

1：加减乘除，取模；

2：快速幂次运算，高精度开平方；

3: 求组合，排列数；

4：取大数末尾0的个数，取大数长度，取大数指定位的数字；

5：可直接将字符串赋值给大数,自动去前导0；

Tips: 读入用scan\_f，输出用print\_f;

\*/

const int DIGIT=4,DEPTH=(int)1e4,big\_N=10010;//四位隔开,即万进制

typedef int bignum\_t[big\_N+1];

char big\_s[big\_N\*DIGIT+2];

int szof=sizeof(bignum\_t);

int comp(const bignum\_t a,const bignum\_t b)

{

int i;

if(a[0]!=b[0])return a[0]-b[0];

for(i=a[0];i;i--)if(a[i]!=b[i])return a[i]-b[i];

return 0 ;

}

int comp(const bignum\_t a,const int b)

{

int c[12]={1};

for(c[1]=b;c[c[0]]>=DEPTH;c[c[0]+1]=c[c[0]]/DEPTH,c[c[0]]%=DEPTH,c[0]++);

return comp(a,c);

}

int comp(const bignum\_t a,const int c,const int d,const bignum\_t b)

{

int i,t=0,O=-DEPTH\*2 ;

if(b[0]-a[0]<d&&c)

return 1;

for(i=b[0];i>d;i--)

{

t=t\*DEPTH+a[i-d]\*c-b[i];

if(t>0)return 1;

if(t<O)return 0;

}

for(i=d;i;i--)

{

t=t\*DEPTH-b[i];

if(t>0)return 1;

if(t<O)return 0;

}

return t>0;

}

void add(bignum\_t a,const bignum\_t b)

{

int i ;

for(i=1;i<=b[0];i++)

if((a[i]+=b[i])>=DEPTH)

a[i]-=DEPTH,a[i+1]++;

if(b[0]>=a[0])

a[0]=b[0];

else

for(;a[i]>=DEPTH&&i<a[0];a[i]-=DEPTH,i++,a[i]++);

a[0]+=(a[a[0]+1]>0);

}

void add(bignum\_t a,const int b)

{

int i=1 ;

for(a[1]+=b;a[i]>=DEPTH&&i<a[0];a[i+1]+=a[i]/DEPTH,a[i]%=DEPTH,i++);

for(;a[a[0]]>=DEPTH;a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);

}

void sub(bignum\_t a,const bignum\_t b)

{

int i ;

for(i=1;i<=b[0];i++)

if((a[i]-=b[i])<0)

a[i+1]--,a[i]+=DEPTH ;

for(;a[i]<0;a[i]+=DEPTH,i++,a[i]--);

for(;!a[a[0]]&&a[0]>1;a[0]--);

}

void sub(bignum\_t a,const int b)

{

int i=1 ;

for(a[1]-=b;a[i]<0;a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH,i++);

for(;!a[a[0]]&&a[0]>1;a[0]--);

}

void sub(bignum\_t a,const bignum\_t b,const int c,const int d)

{

int i,O=b[0]+d ;

for(i=1+d;i<=O;i++)

if((a[i]-=b[i-d]\*c)<0)

a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH ;

for(;a[i]<0;a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH,i++);

for(;!a[a[0]]&&a[0]>1;a[0]--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 大数相乘，读入被乘数a，乘数b，结果保存在c[] \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void mul(bignum\_t c,const bignum\_t a,const bignum\_t b)

{

int i,j ;

memset((void\*)c,0,szof);

for(c[0]=a[0]+b[0]-1,i=1;i<=a[0];i++)

for(j=1;j<=b[0];j++)

if((c[i+j-1]+=a[i]\*b[j])>=DEPTH)

c[i+j]+=c[i+j-1]/DEPTH,c[i+j-1]%=DEPTH ;

for(c[0]+=(c[c[0]+1]>0);!c[c[0]]&&c[0]>1;c[0]--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 大数乘以小数，读入被乘数a，乘数b，结果保存在被乘数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void mul(bignum\_t a,const int b)

{

int i ;

for(a[1]\*=b,i=2;i<=a[0];i++)

{

a[i]\*=b ;

if(a[i-1]>=DEPTH)

a[i]+=a[i-1]/DEPTH,a[i-1]%=DEPTH ;

}

for(;a[a[0]]>=DEPTH;a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);

for(;!a[a[0]]&&a[0]>1;a[0]--);

}

void mul(bignum\_t b,const bignum\_t a,const int c,const int d)

{

int i ;

memset((void\*)b,0,szof);

for(b[0]=a[0]+d,i=d+1;i<=b[0];i++)

if((b[i]+=a[i-d]\*c)>=DEPTH)

b[i+1]+=b[i]/DEPTH,b[i]%=DEPTH ;

for(;b[b[0]+1];b[0]++,b[b[0]+1]=b[b[0]]/DEPTH,b[b[0]]%=DEPTH);

for(;!b[b[0]]&&b[0]>1;b[0]--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 大数相除,读入被除数a，除数b，结果保存在c[]数组 \*/

/\* 需要comp()函数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void div(bignum\_t c,bignum\_t a,const bignum\_t b)

{

int h,l,m,i ;

memset((void\*)c,0,szof);

c[0]=(b[0]<a[0]+1)?(a[0]-b[0]+2):1 ;

for(i=c[0];i;sub(a,b,c[i]=m,i-1),i--)

for(h=DEPTH-1,l=0,m=(h+l+1)>>1;h>l;m=(h+l+1)>>1)

if(comp(b,m,i-1,a))h=m-1 ;

else l=m ;

for(;!c[c[0]]&&c[0]>1;c[0]--);

c[0]=c[0]>1?c[0]:1 ;

}

void div(bignum\_t a,const int b,int&c)

{

int i ;

for(c=0,i=a[0];i;c=c\*DEPTH+a[i],a[i]=c/b,c%=b,i--);

for(;!a[a[0]]&&a[0]>1;a[0]--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 大数平方根，读入大数a，结果保存在b[]数组里 \*/

/\* 需要comp()函数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sqrt(bignum\_t b,bignum\_t a)

{

int h,l,m,i ;

memset((void\*)b,0,szof);

for(i=b[0]=(a[0]+1)>>1;i;sub(a,b,m,i-1),b[i]+=m,i--)

for(h=DEPTH-1,l=0,b[i]=m=(h+l+1)>>1;h>l;b[i]=m=(h+l+1)>>1)

if(comp(b,m,i-1,a))h=m-1 ;

else l=m ;

for(;!b[b[0]]&&b[0]>1;b[0]--);

for(i=1;i<=b[0];b[i++]>>=1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 返回大数的长度 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int length(const bignum\_t a)

{

int t,ret ;

for(ret=(a[0]-1)\*DIGIT,t=a[a[0]];t;t/=10,ret++);

return ret>0?ret:1 ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 返回指定位置的数字，从低位开始数到第b位，返回b位上的数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int digit(const bignum\_t a,const int b)

{

int i,ret ;

for(ret=a[(b-1)/DIGIT+1],i=(b-1)%DIGIT;i;ret/=10,i--);

return ret%10 ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 返回大数末尾0的个数,如果大数为0返回值会出错 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int zeronum(const bignum\_t a)

{

int ret,t ;

for(ret=0;!a[ret+1];ret++);

for(t=a[ret+1],ret\*=DIGIT;!(t%10);t/=10,ret++);

return ret ;

}

void comp(int\*a,const int l,const int h,const int d)

{

int i,j,t ;

for(i=l;i<=h;i++)

for(t=i,j=2;t>1;j++)

while(!(t%j))

a[j]+=d,t/=j ;

}

void convert(int\*a,const int h,bignum\_t b)

{

int i,j,t=1 ;

memset(b,0,szof);

for(b[0]=b[1]=1,i=2;i<=h;i++)

if(a[i])

for(j=a[i];j;t\*=i,j--)

if(t\*i>DEPTH)

mul(b,t),t=1 ;

mul(b,t);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 组合数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void combination(bignum\_t a,int m,int n)

{

int\*t=new int[m+1];

memset((void\*)t,0,sizeof(int)\*(m+1));

comp(t,n+1,m,1);

comp(t,2,m-n,-1);

convert(t,m,a);

delete[]t ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 排列数 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void permutation(bignum\_t a,int m,int n)

{

int i,t=1 ;

memset(a,0,szof);

a[0]=a[1]=1 ;

for(i=m-n+1;i<=m;t\*=i++)

if(t\*i>DEPTH)

mul(a,t),t=1 ;

mul(a,t);

}

#define SGN(x) ((x)>0?1:((x)<0?-1:0))

#define ABS(x) ((x)>0?(x):-(x))

int read(bignum\_t a,int&sgn,char \*buf)

{

char ch;

int i,j;

memset((void\*)a,0,szof),sgn=1;

if(\*buf=='-')sgn=-1,buf++;

for(a[0]=strlen(buf),i=a[0]/2-1;i>=0;i--)

ch=buf[i],buf[i]=buf[a[0]-1-i],buf[a[0]-1-i]=ch ;

for(a[0]=(a[0]+DIGIT-1)/DIGIT,j=strlen(buf);j<a[0]\*DIGIT;buf[j++]='0');

for(i=1;i<=a[0];i++)

for(a[i]=0,j=0;j<DIGIT;j++)

a[i]=a[i]\*10+buf[i\*DIGIT-1-j]-'0' ;

for(;!a[a[0]]&&a[0]>1;a[0]--);

if(a[0]==1&&!a[1])sgn=0 ;

}

void write(const bignum\_t a)

{

int i,j;

for(printf("%d",a[i=a[0]]),i--;i;i--)

for(j=DEPTH/10;j;j/=10)printf("%d",a[i]/j%10);

}

struct bignum

{

bignum\_t num;

int sgn;

inline bignum(){num[0]=1,sgn=0;}

inline int operator!(){return num[0]==1&&!num[1];}

inline bignum&operator=(const bignum&a)//大数赋值

{

memcpy(num,a.num,szof);

sgn=a.sgn ;

return\*this ;

}

inline bignum&operator=(const int a)//int赋值

{

memset(num,0,szof);

num[0]=1 ;

sgn=SGN (a);

add(num,sgn\*a);

return\*this;

}

inline bignum&operator=(char \*ss){read(num,sgn,ss);}//字符串赋值

inline bignum&operator^(const int a){//大数的快速幂

bignum tp;tp=1;int k=a;

while(k){

if(k&1)tp\*=\*this;

k>>=1,\*this\*=(\*this);

}

\*this=tp;

return \*this;

}

inline bignum&operator+=(const bignum&a)

{

if(sgn==a.sgn)add(num,a.num);

else if(sgn&&a.sgn)

{

int ret=comp(num,a.num);

if(ret>0)sub(num,a.num);

else if(ret<0)

{

bignum\_t t ;

memcpy(t,num,szof);

memcpy(num,a.num,szof);

sub (num,t);

sgn=a.sgn ;

}

else memset(num,0,szof),num[0]=1,sgn=0 ;

}

else if(!sgn)

memcpy(num,a.num,szof),sgn=a.sgn ;

return\*this ;

}

inline bignum&operator+=(const int a)

{

if(sgn\*a>0)add(num,ABS(a));

else if(sgn&&a)

{

int ret=comp(num,ABS(a));

if(ret>0)sub(num,ABS(a));

else if(ret<0)

{

bignum\_t t ;

memcpy(t,num,szof);

memset(num,0,szof);

num[0]=1 ;

add(num,ABS (a));

sgn=-sgn ;

sub(num,t);

}

else memset(num,0,szof),num[0]=1,sgn=0 ;

}

else if(!sgn)sgn=SGN(a),add(num,ABS(a));

return\*this ;

}

inline bignum operator+(const bignum&a)

{

bignum ret ;

memcpy(ret.num,num,sizeof (bignum\_t));

ret.sgn=sgn,ret+=a;

return ret ;

}

inline bignum operator+(const int a)

{

bignum ret ;

memcpy(ret.num,num,sizeof (bignum\_t));

ret.sgn=sgn,ret+=a;

return ret ;

}

inline bignum&operator-=(const bignum&a)

{

if(sgn\*a.sgn<0)add(num,a.num);

else if(sgn&&a.sgn)

{

int ret=comp(num,a.num);

if(ret>0)sub(num,a.num);

else if(ret<0)

{

bignum\_t t ;

memcpy(t,num,szof);

memcpy(num,a.num,szof);

sub(num,t),sgn=-sgn;

}else memset(num,0,szof),num[0]=1,sgn=0 ;

}

else if(!sgn)add (num,a.num),sgn=-a.sgn ;

return\*this ;

}

inline bignum&operator-=(const int a)

{

if(sgn\*a<0)add(num,ABS(a));

else if(sgn&&a)

{

int ret=comp(num,ABS(a));

if(ret>0)sub(num,ABS(a));

else if(ret<0)

{

bignum\_t t ;

memcpy(t,num,szof);

memset(num,0,szof);

num[0]=1,add(num,ABS(a)),sub(num,t);

sgn=-sgn;

}else memset(num,0,szof),num[0]=1,sgn=0 ;

}

else if(!sgn)sgn=-SGN(a),add(num,ABS(a));

return\*this ;

}

inline bignum operator-(const bignum&a)

{

bignum ret;

memcpy(ret.num,num,szof);

ret.sgn=sgn,ret-=a;

return ret ;

}

inline bignum operator-(const int a)

{

bignum ret ;

memcpy(ret.num,num,szof);

ret.sgn=sgn,ret-=a;

return ret;

}

inline bignum&operator\*=(const bignum&a)

{

bignum\_t t;

mul(t,num,a.num);

memcpy(num,t,szof);

sgn\*=a.sgn;

return\*this;

}

inline bignum&operator\*=(const int a)

{

mul(num,ABS(a)),sgn\*=SGN(a);

return\*this ;

}

inline bignum operator\*(const bignum&a)

{

bignum ret ;

mul(ret.num,num,a.num);

ret.sgn=sgn\*a.sgn ;

return ret ;

}

inline bignum operator\*(const int a)

{

bignum ret ;

memcpy(ret.num,num,sizeof (bignum\_t));

mul(ret.num,ABS(a)),ret.sgn=sgn\*SGN(a);

return ret ;

}

inline bignum&operator/=(const bignum&a)

{

bignum\_t t ;

div(t,num,a.num);

memcpy (num,t,szof);

sgn=(num[0]==1&&!num[1])?0:sgn\*a.sgn ;

return\*this ;

}

inline bignum&operator/=(const int a)

{

int t ;

div(num,ABS(a),t);

sgn=(num[0]==1&&!num [1])?0:sgn\*SGN(a);

return\*this ;

}

inline bignum operator/(const bignum&a)

{

bignum ret ;bignum\_t t;

memcpy(t,num,szof);

div(ret.num,t,a.num);

ret.sgn=(ret.num[0]==1&&!ret.num[1])?0:sgn\*a.sgn ;

return ret ;

}

inline bignum operator/(const int a)

{

bignum ret;int t;

memcpy(ret.num,num,szof);

div(ret.num,ABS(a),t);

ret.sgn=(ret.num[0]==1&&!ret.num[1])?0:sgn\*SGN(a);

return ret ;

}

inline bignum&operator%=(const bignum&a)

{

bignum\_t t ;

div(t,num,a.num);

if(num[0]==1&&!num[1])sgn=0 ;

return\*this ;

}

inline int operator%=(const int a)

{

int t ;div(num,ABS(a),t);

memset(num,0,sizeof (bignum\_t));

num[0]=1,add(num,t);

return t ;

}

inline bignum operator%(const bignum&a)

{

bignum ret;

bignum\_t t;

memcpy(ret.num,num,szof);

div(t,ret.num,a.num);

ret.sgn=(ret.num[0]==1&&!ret.num [1])?0:sgn ;

return ret ;

}

inline int operator%(const int a)

{

bignum ret;int t;

memcpy(ret.num,num,szof);

div(ret.num,ABS(a),t);

memset(ret.num,0,szof);

ret.num[0]=1 ;

add(ret.num,t);

return t ;

}

inline bignum&operator++(){\*this+=1;return\*this;}

inline bignum&operator--(){\*this-=1;return\*this;}

inline int operator>(const bignum&a)

{

return sgn>0?(a.sgn>0?comp(num,a.num)>0:1):(sgn<0?(a.sgn<0?comp(num,a.num)<0:0):a.sgn<0);

}

inline int operator>(const int a)

{

return sgn>0?(a>0?comp(num,a)>0:1):(sgn<0?(a<0?comp(num,-a)<0:0):a<0);

}

inline int operator>=(const bignum&a)

{

return sgn>0?(a.sgn>0?comp(num,a.num)>=0:1):(sgn<0?(a.sgn<0?comp(num,a.num)<=0:0):a.sgn<=0);

}

inline int operator>=(const int a)

{

return sgn>0?(a>0?comp(num,a)>=0:1):(sgn<0?(a<0?comp(num,-a)<=0:0):a<=0);

}

inline int operator<(const bignum&a)

{

return sgn<0?(a.sgn<0?comp(num,a.num)>0:1):(sgn>0?(a.sgn>0?comp(num,a.num)<0:0):a.sgn>0);

}

inline int operator<(const int a)

{

return sgn<0?(a<0?comp(num,-a)>0:1):(sgn>0?(a>0?comp(num,a)<0:0):a>0);

}

inline int operator<=(const bignum&a)

{

return sgn<0?(a.sgn<0?comp(num,a.num)>=0:1):(sgn>0?(a.sgn>0?comp(num,a.num)<=0:0):a.sgn>=0);

}

inline int operator<=(const int a)

{

return sgn<0?(a<0?comp(num,-a)>=0:1):

(sgn>0?(a>0?comp(num,a)<=0:0):a>=0);

}

inline int operator==(const bignum&a)

{

return(sgn==a.sgn)?!comp(num,a.num):0 ;

}

inline int operator==(const int a)

{

return(sgn\*a>=0)?!comp(num,ABS(a)):0 ;

}

inline int operator!=(const bignum&a)

{

return(sgn==a.sgn)?comp(num,a.num):1 ;

}

inline int operator!=(const int a)

{

return(sgn\*a>=0)?comp(num,ABS(a)):1 ;

}

inline int operator[](const int a){return digit(num,a);}

friend inline bignum sqrt(const bignum&a)//大数开平方

{

bignum ret;

bignum\_t t;

memcpy(t,a.num,szof);

sqrt(ret.num,t);

ret.sgn=ret.num[0]!=1||ret.num[1];

return ret ;

}

inline int length(){return :: length(num);}

inline int zeronum(){return :: zeronum(num);}

inline bignum C(const int m,const int n)//大数存组合数

{

combination(num,m,n);

sgn=1;

return\*this ;

}

inline bignum P(const int m,const int n)//大数存排列数

{

permutation(num,m,n);

sgn=1;

return\*this ;

}

void scan\_f(){scanf("%s",big\_s);read(num,sgn,big\_s);}//大数读入

void print\_f(){if(sgn<0)printf("-");write(num);}//大数输出

};

bignum A,B;

int main(){

A.scan\_f();

A.print\_f();puts("");

return 0 ;

}