超级高精度 加减乘

```cpp
#include<iostream>
#include<string>
#include<cstring>
#include<cstdio>
using namespace std;
const int BIT = 2005;
const int N = BIT*BIT;
struct bign
{
    int len,s[N];
    bign()  {  memset(s,0,sizeof(s));  len=1;  }
    bign(int num)  {  *this=num; }
    bign(char *num) { *this=num; }
    bign operator =(int num)
    {
        char c[N];
        sprintf(c,"%d",num);
        *this=c;
        return *this;
    }
    bign operator =(const char *num)
    {
        len=strlen(num);
        for (int i=0;i<len;i++) s[i]=num[len-1-i]-'0';
        return *this;
    }
    string str()
    {
        string res="";
        for (int i=0;i<len;i++) res=(char)(s[i]+'0')+res;
        return res;
    }
    void clean()
    {
        while (len>1&&!s[len-1]) len--;
    }
    bign operator +(const bign &b)
    {
        bign c;
        c.len=0;
        for (int i=0,g=0;g||i<len||i<b.len;i++)
        {
            int x=g;
            if (i<len) x+=s[i];
            if (i<b.len) x+=b.s[i];
            c.s[c.len++]=x%10;
            g=x/10;
        }
        return c;
    }
    bign operator -(const bign &b)
    {
        bign c;
        c.len=0;
        int x;
        for (int i=0,g=0;i<len;i++)
        {
            x=s[i]-g;
            if (i<b.len) x-=b.s[i];
            if (x>=0) g=0;
            else{
                x+=10;
                g=1;
            };
            c.s[c.len++]=x;
        }
        c.clean();
        return c;
    }
    bign operator *(const bign &b)
    {
        bign c;
        c.len=len+b.len;
        for (int i=0;i<len;i++) for (int j=0;j<b.len;j++)
  c.s[i+j]+=s[i]*b.s[j];
        for (int i=0;i<c.len-1;i++) { c.s[i+1]+=c.s[i]/10; c.s[i]%=10; }
        c.clean();
        return c;
    }
    bool operator <(const bign &b)
    {
        if (len!=b.len) return len<b.len;
        for (int i=len-1;i>=0;i--)
             if (s[i]!=b.s[i]) return s[i]<b.s[i];
        return false;
    }
    bool operator ==(const bign &b)
    {
        if (len!=b.len) return false;
        for (int i=len-1;i>=0;i--)
             if (s[i]!=b.s[i]) return false;
        return true;
    }
    bool operator !=(const bign &b)
    {
        if (len!=b.len) return true;
        for (int i=len-1;i>=0;i--)
             if (s[i]!=b.s[i]) return true;
        return false;
    }
    bign operator +=(const bign &b)
    {
        *this=*this+b;
        return *this;
    }
    bign operator -=(const bign &b)
    {
        *this=*this-b;
        return *this;
    }
};
istream& operator >>(istream &in,bign &x)
{
  string s;
  in>>s;
  x=s.c_str();
  return in;
}
ostream& operator <<(ostream &out,bign &x)
{
    out<<x.str();
    return out;
}
int main(){
    bign a,b,c;
    //ios::sync_with_stdio(false);
    cin>>a>>b;
    c=a*b;
    cout<<c<<endl;
    return 0;
}
```

高精度加减乘 高-低除

```cpp
#include<iostream>
#include<cstring>
using namespace std;

string prec_plus(string plus_s1,string plus_s2){
    int plus_i1[10100],plus_i2[10100];
    int l1=plus_s1.length(),l2=plus_s2.length();
    string ans="";
    int len=max(l1,l2);
    memset(plus_i1,0,sizeof(plus_i1));
    memset(plus_i2,0,sizeof(plus_i2));
    for(int i=l1-1;i>=0;i--)
        plus_i1[l1-i-1]=plus_s1[i]-'0';
    for(int i=l2-1;i>=0;i--)
        plus_i2[l2-i-1]=plus_s2[i]-'0';
    for(int i=0;i<len;i++){
        plus_i1[i]+=plus_i2[i];
        plus_i1[i+1]+=plus_i1[i]/10;
        plus_i1[i]%=10;
    }
    if(plus_i1[len]!=0) len++;
    while(plus_i1[len-1]==0 and len>1)
        len--;
    for(int i=len-1;i>=0;i--)
```

```
        ans=ans+char(plus_i1[i]+'0');
    return ans;
}

string prec_minus(string minus_s1,string minus_s2){
    int minus_i1[10100],minus_i2[10100];
    int l1=minus_s1.length(),l2=minus_s2.length();
    string ans="";
    int len=max(l1,l2);
    memset(minus_i1,0,sizeof(minus_i1));
    memset(minus_i2,0,sizeof(minus_i2));
    for(int i=l1-1;i>=0;i--)
        minus_i1[l1-i-1]=minus_s1[i]-'0';
    for(int i=l2-1;i>=0;i--)
        minus_i2[l2-i-1]=minus_s2[i]-'0';
    for(int i=0;i<len;i++){
        minus_i1[i]-=minus_i2[i];
        if(minus_i1[i]<0){
            minus_i1[i]+=10;
            minus_i1[i+1]--;
        }
    }
    while(minus_i1[len-1]==0 and len>1)
        len--;
    for(int i=len-1;i>=0;i--)
        ans=ans+char(minus_i1[i]+'0');
    return ans;
}

string prec_multiply(string multiply_s1,string multiply_s2){
    int multiply_i1[1010],multiply_i2[1010],multiply_i3[1010];
    int l1=multiply_s1.length(),l2=multiply_s2.length();
    string ans="";
    int len=(l1+l2);
    memset(multiply_i1,0,sizeof(multiply_i1));
    memset(multiply_i2,0,sizeof(multiply_i2));
    memset(multiply_i3,0,sizeof(multiply_i3));
    for(int i=l1-1;i>=0;i--)
        multiply_i1[l1-i-1]=multiply_s1[i]-'0';
    for(int i=l2-1;i>=0;i--)
        multiply_i2[l2-i-1]=multiply_s2[i]-'0';
    for(int i=0;i<l1;i++){
        for(int j=0;j<l2;j++){
            multiply_i3[i+j]+=multiply_i1[i]*multiply_i2[j];
            multiply_i3[i+j+1]+=multiply_i3[i+j]/10;
            multiply_i3[i+j]%=10;
        }
    }
    while(multiply_i3[len-1]==0 and len>1)
        len--;
    for(int i=len-1;i>=0;i--)
        ans=ans+char(multiply_i3[i]+'0');
    return ans;
}
```

```
string prec_division(string div_s1,int div_i2){
    int div_i1[10100];
    memset(div_i1,0,sizeof(div_i1));
    int l1=div_s1.length();
    for(int i=0;i<l1;i++)
        div_i1[i]=div_s1[i]-'0';
    int div_t=0;
    for(int i=0;i<l1;i++){
        div_t=div_t*10+div_i1[i];
        div_i1[i]=div_t/div_i2;
        div_t%=div_i2;
    }
    bool div_f=false;
    string ans;
    for(int i=0;i<l1;i++){
        if(div_i1[i]) div_f=true;
        if(div_f or i==l1-1) ans=ans+char(div_i1[i]+'0');
    }
    return ans;
}
```

归并排序

```cpp
#pragma GCC optimize(2)
#include<bits/stdc++.h>
#define abss(x) ((x)>(0)?(x):(-1)*(x))
#define maxs(a,b) ((a)>(b)?(a):(b))
#define mins(a,b) ((a)<(b)?(a):(b))
#define FOR(i,a,b) for(register int i=(a);i<=(b);i++)
#define ROF(i,a,b) for(register int i=(a);i>=(b);i--)
#define mem(a) memset(a,0,sizeof(a))
const int INF (1<<30);
const int inf (-1<<30);
using namespace std;

int tmp[int(1e5)]={};
void merge_sort(int q[],int l,int r){

    if(l>=r) return;
    int mid=l+r>>1;
    merge_sort(q,l,mid);
    merge_sort(q,mid+1,r);
    int k=0,i=l,j=mid+1;
    while(i<=mid and j<=r){
        if(q[i]<=q[j]) tmp[k++]=q[i++];
        else tmp[k++]=q[j++];
    }
    while(i<=mid) tmp[k++]=q[i++];
    while(j<=r) tmp[k++]=q[j++];
    for(i=l,j=0;i<=r;i++,j++) q[i]=tmp[j];
}
```

```
int main(){
    int n,a[int(1e5)];
    cin>>n;
    FOR(i,0,n-1) scanf("%d",a+i);
    merge_sort(a,0,n-1);
    FOR(i,0,n-1) printf("%d ",a[i]);
    return 0;
}
```

快速排序

```cpp
#pragma GCC optimize(2)
#include<bits/stdc++.h>
#define abss(x) ((x)>(0)?(x):(-1)*(x))
#define maxs(a,b) ((a)>(b)?(a):(b))
#define mins(a,b) ((a)<(b)?(a):(b))
#define FOR(i,a,b) for(register int i=(a);i<=(b);i++)
#define ROF(i,a,b) for(register int i=(a);i>=(b);i--)
#define mem(a) memset(a,0,sizeof(a))
const int INF (1<<30);
const int inf (-1<<30);
using namespace std;

void qsort(int a[],int l,int r){
    int mid=a[(l+r)/2];
    int i=l,j=r;
    while(i<j){
        while(a[i]<mid) i++;
        while(a[j]>mid) j--;
        if(i<=j){
            swap(a[i],a[j]);
            i++;j--;
        }
    }
    if(l<j) qsort(a,l,j);
    if(r>i) qsort(a,i,r);
}

int main(){
    int n,a[int(1e5)];
    cin>>n;
    FOR(i,0,n-1) scanf("%d",a+i);
    qsort(a,0,n-1);
    FOR(i,0,n-1) printf("%d ",a[i]);
    return 0;
}

```