实训模板库 1——高精度

注 1：模板中 L 为每个数的最大长度，如果题目中要求位数超过 2000 位，需要如下修改：将 const int L=2000;中的 2000 修改为题目中要求的最大长度

注 2：代码基于 c++的 string 类型，建议先学习一些 string 的基本操作。

一、高精度加法

注：代码要求 a，b 都是非负整数。

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int L=1000;
string add(string a,string b)
{
    string ans;
    int na[L]={0},nb[L]={0};
    int la=a.size(),lb=b.size();
    for(int i=0;i<la;i++)
        na[la-1-i]=a[i]-'0';
    for(int i=0;i<lb;i++)
        nb[lb-1-i]=b[i]-'0';
    int lmax=la>lb?la:lb;
    for(int i=0;i<lmax;i++)
        na[i]+=nb[i],na[i+1]+=na[i]/10,na[i]%=10;
    if(na[lmax]) lmax++;
    for(int i=lmax-1;i>=0;i--)
        ans+=na[i]+'0';
    return ans;
}
int main()
{
    string a,b;
    while(cin>>a>>b)
        cout<<add(a,b)<<endl;
    return 0;
}
```

二、高精度减法

注：代码要求 a，b 都是非负整数，且大数减小数。

```cpp
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
const int L=110;
string sub(string a,string b) //要求 a>b
{
    string ans;
    int na[L]={0},nb[L]={0};
    int la=a.size(),lb=b.size();
    for(int i=0;i<la;i++)
        na[la-1-i]=a[i]-'0';
    for(int i=0;i<lb;i++)
        nb[lb-1-i]=b[i]-'0';
    int lmax=la>lb?la:lb;
    for(int i=0;i<lmax;i++)
    {
        na[i]-=nb[i];
        if(na[i]<0) na[i]+=10,na[i+1]--;
    }
    while(!na[--lmax]&&lmax>0);
        lmax++;
    for(int i=lmax-1;i>=0;i--)
        ans+=na[i]+'0';
    return ans;
}
int main()
{
    string a,b;
    while(cin>>a>>b) cout<<sub(a,b)<<endl;
    return 0;
}
```

三、高精度乘法

注：代码要求 a，b 都是非负整数。

```
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
const int L=110;
string mul(string a,string b)
{
    string s;
    int na[L],nb[L],nc[L],La=a.size(),Lb=b.size();
    fill(na,na+L,0);
    fill(nb,nb+L,0);
    fill(nc,nc+L,0);
    for(int i=La-1;i>=0;i--)
        na[La-i]=a[i]-'0';
    for(int i=Lb-1;i>=0;i--)
        nb[Lb-i]=b[i]-'0';
    for(int i=1;i<=La;i++)
        for(int j=1;j<=Lb;j++)
            nc[i+j-1]+=na[i]*nb[j];
    for(int i=1;i<=La+Lb;i++)
        nc[i+1]+=nc[i]/10,nc[i]%=10;
    if(nc[La+Lb])
        s+=nc[La+Lb]+'0';
    for(int i=La+Lb-1;i>=1;i--)
        s+=nc[i]+'0';
    while (s[0]=='0') s=s.substr(1);  //注意：a,b 有一个为 0 的情况
    if (!s.size()) s="0";
    return s;
}
int main()
{
    string a,b;
    while(cin>>a>>b) cout<<mul(a,b)<<endl;
    return 0;
}
```

四、高精度除以高精度

注：代码要求 a，b 都是非负整数。

```cpp
#include<iostream>
#include<algorithm>
using namespace std;
string div(string a,long long b) //高精度 a 除以单精度 b
{
    //b 最好用 long long
    string r,ans;
    long long d=0;
    if(a=="0") return a;
    for(int i=0;i<a.size();i++)
    {
        r+=(d*10+a[i]-'0')/b+'0';
        d=(d*10+(a[i]-'0'))%b;
    }
    int p=0;
    for(int i=0;i<r.size();i++)
    if(r[i]!='0') {p=i;break;}
    return r.substr(p);
}
int main()
{
    string a;
    long long b;
    while(cin>>a>>b)  cout<<div(a,b)<<endl;
    return 0;
}
```

补充：十六进制加法（EOJ 3037）
题目大意：输入两个十六进制表示的整数，以十六进制输出两个数的和。
注 1：和十进制加法区别在于 decode 和 encode 函数，以及进位标准
注 2：代码中使用 reverse 函数，反转字符串，使得每一位对齐

```
#include <cstring>
#include <string>
#include <algorithm>
#include <iostream>
using namespace std;
int decode(char c)
{
    int code=0;
    if (isdigit(c))
        code=c-'0';
    else if (isalpha(c))
        code=c-'A'+10;
    return code;
}
string encode(int c)
{
    string s=" ";
    if (c>=10) s[0]=c-10+'A';
    else s[0]=c+'0';
    return s;
}
string solve(string s1,string s2)
{
    string s="";
    reverse(s1.begin(),s1.end());
    reverse(s2.begin(),s2.end());
    int maxl=max(s1.length(),s2.length())+1;
    int cc=0;
    for (int i=0;i<maxl;i++)
    {
        int p1=decode(s1[i]),p2=decode(s2[i]);
        if (i>=s1.length()) p1=0;
        if (i>=s2.length()) p2=0;
        s+=encode((p1+p2+cc)%16);
        cc=(p1+p2+cc)/16;
    }
    while (s[s.length()-1]=='0')
        s=s.substr(0,s.length()-1);
    reverse(s.begin(),s.end());
    if (!s.length()) s="0";
```

```
41.         return s;
42.     }
43.     int main()
44.     {
45.         int t,cas=0;
46.         string s1,s2;
47.         cin>>t;
48.         while (t--)
49.         {
50.             cin>>s1>>s2;
51.             printf("case #%d:\n",cas++);
52.             cout<<solve(s1,s2)<<endl;
53.         }
54.         return 0;
55.     }
```

实训模板库 2——数与数论

一、同余的性质

算法分析：$(x+y)\equiv((x \bmod m)+(y \bmod m))(\bmod m)$，也可以写成 C 语言表达式：

$(x+y)\%m=((x\%m)+(y\%m))\%m$。

类似有乘法的表达式：$(x*y)\%m=((x\%m)*(y\%m))\%m$。

减法表达式略有不同，考虑到减完会有负数：$(x-y)\%m=((x\%m)-(y\%m)+m)\%m$

例子：求 $n!(\bmod m)\ |1<=n<=2*10^6,1<=m<=10^9$

算法分析：采用边乘边取模的思想，而不是算完再取模

```
#include <iostream>
using namespace std;
long long fac_mod(int n,long long m)
{
    long long ans=1;
    for (int i=1;i<=n;i++)
    {
        ans*=i;
        ans%=m;
    }
    return ans%m;  //n=0，m=1 时，返回 0
}
int main()
{
    int n;
    long long m;
    while (cin>>n>>m)
        cout<<fac_mod(n,m)<<endl;
    return 0;
}
```

二、快速幂取模

问题描述：求 $x^y(\bmod m)$，其中 mod 表示取余。

算法分析：快速幂算法依赖于以下明显的结论。

$x^y(\bmod m)=((x^2)^{y/2})(\bmod m)$ |y是偶数

$x^y(\bmod m)=((x^2)^{y/2}*x)(\bmod m)$ |y是奇数 ($y/2$表示y除2下取整)

有了上述两个公式后，我们可以得出以下的结论：

1.如果 b 是偶数，我们可以记 $k=x^2(\bmod m)$，那么求 $x^{y/2}(\bmod m)$ 就可以了。

2.如果 b 是奇数，我们也可以记 $k=x^2(\bmod m)$，那么求 $(x^{y/2}*x)(\bmod m)$ 就可以了。

```
#include <iostream>
using namespace std;
long long power_ntt(long long a,long long b,long long c)
{
    long long ans=1;
    a=a%c;
    while(b>0)
    {
        if(b%2==1)
        ans=(ans*a)%c;
        b=b/2;
        a=(a*a)%c;
    }
    return ans%c;   //b=0，c=1 时，返回 0
}
int main()
{
    long long a,b,c;
    while (cin>>a>>b>>c)
    {
        //find (a^b)%c
        cout<<power_ntt(a,b,c);
    }
    return 0;
}
```

三、判断素数

问题描述：判断一个数 n 是否为素数

算法分析：只要判断从 2 开始到$\sqrt{n}$每个数都不能被 n 整除即可。

```
#include <iostream>
#include <cmath>
using namespace std;
bool is_prime(long long n)
{
    if (n==1) return false;
    for (long long i=2;i<=sqrt(n);i++)
        if (n%i==0) return false;
    return true;
}
int main()
{
    long long n;
    while (cin>>n)
        cout<<is_prime(n)<<endl;
    return 0;
}
```

问题补充：求一个数 n 的所有小于 n 的约数和（注意平方数的处理）

```
#include <iostream>
#include <cmath>
using namespace std;
long long sum_divisor(long long n)
{
    if (n==1) return 0;
    long long sum=1;
    for (long long i=2;i<=sqrt(n);i++)
        if (n%i==0)
        {
            sum+=i;
            if (i*i!=n) sum+=n/i;
        }
    return sum;
}
int main()
{
    long long n;
    while (cin>>n)
        cout<<sum_divisor(n)<<endl;
    return 0;
}
```

四、欧拉筛法：

问题描述：求 1-maxn 中所有的素数

代码注释：完成 get_prime 函数后，is_prime 这个 vector 中存的就是所有素数

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <cstring>
#define maxn 10000+5
using namespace std;
vector <int> is_prime;
void get_prime()
{
    is_prime.clear();
    bool prime[maxn+5];
    memset(prime,true,sizeof(prime));
    is_prime.push_back(2);
    for (int i=3;i<=maxn;i+=2)
    {
        if (prime[i])
        {
            is_prime.push_back(i);
            for (int j=2*i;j<=maxn;j+=i)
                prime[j]=false;
        }
    }
    return;
}
int main()
{
    get_prime();
    for (int i=0;i<is_prime.size();i++)
        cout<<is_prime[i]<<' ';
    cout<<endl;
    return 0;
}
```

五、最小公倍数与最大公约数

问题描述：求 gcd(a,b)和 lcm(a,b)

算法分析：基于欧拉公式 gcd(a,b)=gcd(a,a%b)

```
#include <iostream>
using namespace std;
long long gcd(long long a,long long b)
{
    return (b==0)?a:gcd(b,a%b);
}
long long lcm(long long a,long long b)
{
    return a/gcd(a,b)*b;
    //should not be return a*b/gcd(a,b);
}
int main()
{
    long long m,n;
    while (cin>>n>>m)
        cout<<gcd(m,n)<<' '<<lcm(m,n)<<endl;
    return 0;
}
```

实训模板库 3——string 类型

一、头文件

```
#include <string>
using namespace std;
```

二、string 比较

分析：string 类模板中重载了运算符">","<",">=","<=","!=","==",所以可以像整数一样进行比较，比较规则：字典序。

例："abcd"!="abc","ab"<"b","ab"=="ab"。以上比较均返回 1。

三、string 连接

分析：string 重载了"+","+="运算符，功能类似于 C 语言里面的 strcat 函数。

```
string s1="ab",s2="cd";
s=s1+s2;  //s="abcd"
s+=s;  //s="abcdabcd"
```

四、string 和 char *的转换

①char *转 string

```
char t[maxl];
strcpy(t,"abcd");  //t="abcd"
string s=t;   //s="abcd"
```

②string 转 char *

```
char t[maxl];
string s="abcd";
t=s.c_str();
```

例：将 string 转成 long long（假设 string 中存放数字）

分析：C 语言中有 atoll 函数将字符串数组转成 long long。

```
string s="123456789";
long long n=atoll(s.c_str()); //将 string 转成字符串数组，才可使用 atoll 函数
```

五、string 子串

分析：函数 string substr(int pos=0,int n=npos);返回 pos 开始的 n 个字符组成的字符串。

```
string s="123456789";
string sub=s.substr(1,4); //s="2345"
```

六、string 长度

分析：函数 size_t length();返回字符串的长度。

注意：size_t 类型是无符号数，所以 1-2 在无符号中会造成负溢出现象。

例：求字符串 a 和 b 的长度之差：(int) (a.length()-b.length())

七、string 查找函数

分析：两个函数 int find(const string &s);从第一个位置开始，从左向右查找字符串 s 在当前串中的位置，int rfind(const string &s);从最后一个字符位置开始，从右向左查找字符串 s 在当前串中的位置，返回值：如果成功找到，返回所在位置（int 类型），如果没有找到，返回 string::npos。

```
string s,t;
if (s.find(t)!=string::npos)
    cout<<s.find(t)<<endl;   //s 中找到了 t
else cout<<"not found."<<endl;
```

八、string 插入函数
分析：函数 string &insert(int p0,const string &s);可以简单实现在 pos 位置插入 string 类型字符串 s。

```
string a="abcd";
string b="xyz";
a.insert(1,b);  //a="axyzbcd"
```

九、string 迭代器
分析：begin()返回 string 类型 s 第一个位置的迭代器，end()返回 string 类型 s 最后一个位置的迭代器。
例：反转一个字符串，利用 reverse 函数。

```
string s="abcd";
reverse(s.begin(),s.end());  //s="dcba"
```

十、string 删除函数
分析：函数 iterator erase(iterator first, iterator last);删除[first，last）之间的所有字符，返回删除后迭代器的位置。
十一、string 读入
①空格为结束标志：cin>>s;，注意 cin 不读入"\n"和" "
②读入一行数据：getline(cin,s);，注意 getline 自动过滤"\n"
十二、字符串流处理
①头文件

```
#include <sstream>
```

②一个例子

```
string input="hello,this is a test";
istringstream is(input);  //istringstream 字符串读入流
string s1,s2,s3,s4;
is>>s1>>s2>>s3>>s4;//s1="hello,this",s2="is",s3="a",s4="test"
ostringstream os;  //ostringstream 字符串输出流
os<<s1<<" "<<s2<<" "<<s3<<" "<<s4;
cout<<os.str();  //输出 hello,this is a test
```

③特点分析：字符串当读到空格就会断开，类型于 scanf 读入字符串
④应用：使用 STL 的 set 集合类模板可以实现统计单词个数。

实训模板库 4——数据排序

一、头文件

```
#include <algorithm>
using namespace std;
```

二、int,long long,double,string 类型数组的排序（以 long long 为例）

```
#include <iostream>
#include <algorithm>
#define maxn 10000+5
using namespace std;
bool cmp(long long a,long long b)
{
    return a<b;
}
long long a[maxn];
int main()
{
    int n;
    cin>>n;
    for (int i=0;i<n;i++)
        cin>>a[i];
    sort(a,a+n,cmp);
    for (int i=0;i<n;i++)
        cout<<a[i]<<' ';
    cout<<endl;
    return 0;
}
```

note：由于 string 类型中重载了<运算符，所以 string 类型数组可以类似 long long 按照字典序排序。

三、含有 double 类型的结构体排序

分析：由于 double 存在精度问题，所以要让两个 double 类型数组相等是很难的事情，一般的题目中，只要两个 double 的数的差的绝对值小于 1e-7，就认为这两个 double 类型的数相等，要求按 first 升序排序，first 相等时按 second 降序排序

```
#include <iostream>
#include <algorithm>
#include <cmath>
#define maxn 10000+5
#define eps 1e-7
using namespace std;
struct node
{
    double first,second;
};
bool cmp(node a,node b)
{
    if (fabs(a.first-b.first)>eps)
        return a.first<b.first;
    return a.second>b.second;
}
node a[maxn];
int main()
{
    int n;
    cin>>n;
    for (int i=0;i<n;i++)
        cin>>a[i].first>>a[i].second;
    sort(a,a+n,cmp);
    for (int i=0;i<n;i++)
        cout<<a[i].first<<' '<<a[i].second<<endl;
    return 0;
}
```

四、可能会遇到的两类排序题

（1）这就是一道排序题

特点分析：这类题目的特点是你一看题目就知道这是一道排序题，比如我们熟悉的成绩排序，文献排序这些题目，这类题目的特点是：题目比较啰嗦，比较规则也比较复杂。

算法分析：首先，我们应该理清题目的思路，包括：有哪些比较规则，对于数据要做哪些处理，然后，我们把每个数据需要的信息放在一个结构体里面（我们理解为“信息的捆绑”），一般这个结构体里包含两类数据：原始的读入数据（用于最后的输出），处理后的一些信息（用于之后的排序作为比较规则），最后按照要求排序（注意有多个比较规则时候的比较函数写法，详见前面的结构体排序比较函数写法）。

例题分析：

OJ 2994 数组排序

给定一个长度为 N 的整数数组，按如下规则排序并输出。首先，输出重复出现次数最多的元素；出现次数相同时，按元素值由小到大输出。例如对于数组 1 2 3 3 4 2 3 1 5 7 排序后输出 3 3 3 1 1 2 2 4 5 7。

Input

第 1 行：一个整数 T（1≤T≤10）为问题数。

对于每组测试数据：

第 1 行是一个正整数：数组长度 N (1 < N < 2000)；

第 2 行有 N 个整数：分别为第 1 至第 N 个元素的值 a1,a2,…,aN

(对所有 i, 0≤ai<500)。任意两个整数之间由一个空格分开。

Output

对于每个问题，输出一行问题的编号（0 开始编号，格式：case #0: 等）。

然后在一行中输出排序后的数组元素，每两个整数之间由一个空格分开，最后一个整数后面没有空格。

算法分析：题目中明确说明了排序规则：数据出现的次数，数据的大小。于是我们只要用一个结构体要这两个信息“捆绑在一起”就可以了，具体来说，在排序之前，先要预处理有多少个不同的数据，每个数据出现的次数，把信息处理完后，按照要求排序，输出就可以啦，至于输出方法，留给读者思考，以下给出排序部分的代码：

```
1.    struct node
2.    {
3.        int n,cnt;
4.    };
5.    bool cmp(node a,node b)
6.    {
7.        if (a.cnt!=b.cnt) return a.cnt>b.cnt;
8.        return a.n<b.n;
9.    }
```

note：代码中 n 表示数据大小，cnt 表示数据出现次数。

（2）这竟然也是排序题

特点分析：这类题目一般题目很简单，而且看上去也排序没有关系，但是将问题稍微转换，即可变成简单的排序题。

算法分析：如果数据范围不大的话，也可以暴力求解（术语我们称为“模拟”）。

例题：

OJ 3003 最小向量点积

两个向量 a= [a1, a2,…, an] 和 b= [b1, b2,…, bn] 的点积定义为：

$$\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_i=a_1b_1+a_2b_2+\cdots+a_nb_n$$

例如，两个三维向量 [1, 3, -5] 和 [4, -2, -1] 的点积是

$$\begin{bmatrix}1 & 3 & -5\end{bmatrix}\cdot\begin{bmatrix}4 & -2 & -1\end{bmatrix}=(1)(4)+(3)(-2)+(-5)(-1)=3$$

假设允许对每个向量中的坐标值进行重新排列。找出所有排列中点积最小的一种排列，输出最小的那个点积值。上例中的一种排列 [3 1 -5] 和 [-2 -1 4] 的点积为-27，这是最小的点积。

Input

第 1 行：一个整数 T (1≤T≤10) 为问题数。

接下来每个问题有 3 行。第 1 行是一个整数 n (1≤n≤1000)，表示两个向量的维数。第 2 行和第 3 行分别表示向量 a 和向量 b。每个向量都有 n 个由一个空格分隔的坐标值 (-1000≤坐标值≤1000) 组成。

Output

对于每个问题，输出一行问题的编号（0 开始编号，格式：case #0: 等）。

然后对应每个问题在一行中输出最小点积值。

算法分析：我们可以用数学模型描述一下这道题目：

给定一个数列$\{a_n\}=\{a_1,a_2,...,a_n\}$，数列$\{b_n\}=\{b_1,b_2,...,b_n\}$，求一个 1-n 的排列$\{j_n\}=\{j_1,j_2,...,j_n\}$，其中$(1<=j_i<=n$，且 j_i 互不相同)，使得 $a_1*b_{j1}+a_2*b_{j2}+...a_n*b_{jn}$ 最小，这让我们可能会想到排序不等式，排序不等式告诉我们当 a 数组升序，b 数组降序或者顺序相反时，达到最小值（当 a 和 b 同时升序，或降序时达到最大值）。于是问题变成将 a 数组从小到大排序，b 数组从大到小排序，然后对应数乘起来求和即可，注意答案用 long long 保存。

实训模板库 5——其他问题

Q：头文件太多怎么办？

A：c++里面有一个包含所有头文件的“万能头文件”：

```
#include <bits/stdc++.h>
```

只要一句话就包含所有头文件

Q：为什么用 c++函数库的函数报错？

A：有三种可能：

①头文件没有 include，解决可以 include“万能函数库”。

②没有写 using namespace std;

③库函数使用有错误。

Q：提交之后几种错误类型以解决方法?

A:①wrong answer

如果所有数据都 WA 的话，可能是算法错误，但也有可能是个数错误，如“case #0:”有没有写错，排除这些之后，就是算法错误。如果部分数据 WA 的话，可能是一些特殊数据有问题，可以不妨尝试最大的数和最小的数。

②runtime error

一般有三种可能：第一是数组越界，一般比如题目给的最大数据是 10000，我们会多开几个空间，来确保不出现数组下标越界，如：int a[10005]，第二种是对空指针取值，一般解决是在取值前先判断指针是否存在，如：if(p && *p==0)。第三种是算术错误，比如除数是 0。

③time limit exceeded

一般有两种可能：第一种是数据读完，但没有结束程序，如：while (scanf(“%d”,&n))就会造成读完数据一直没有结束程序，解决有两种：while (scanf(“%d”,&n)!=EOF)或者 while (cin>>n)。第二种是死循环，或者循环太多，前者可能是算法上有比较大的问题，后者一般会得到部分分，可以考虑放弃剩下的数据，继续做后面的题目。

Q：实训的规则是什么？

A：①实训可以用的语言：C 和 C++

②计分规则：基于 COI 的计分规则，一个测试数据 10 分，过一个数据得 10 分，一道题目 10 个测试数据，全对得 100 分。（如果不是 10 个测试数组，则每个测试数据平分 100 分）