实训模板库 1——高精度

注：模板中 maxl 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#define maxl 2000 中的 2000 修改为题目中要求的最大长度

一、高精度加法

```
#include <stdio.h>
#include <string.h>
#define maxl 2000
#define max(a,b) (a>b)?(a):(b)
//maxl is the max length of the number
//should be changed according to the problem
using namespace std;
void add(char *s,char *t,char *u)
{
    /*pre condition:s and t are two numbers
                    every character in s and t contain only '0'-'9'
                    s and t should be positive numbers
      post condition:u is s+t
                     make sure u has a length smaller than maxl
    */
    int i;
    int lens=strlen(s),lent=strlen(t);
    int k=max(lens,lent)+1;
    for (i=0;i<lens;i++)
        s[i]-='0';
    for (i=0;i<lent;i++)
        t[i]-='0';
    int cc=0,p=lens-1,q=lent-1;
    for (i=k;i>=0;i--)
    {
        int x=cc;
        if (p>=0) x+=s[p],p--;
        if (q>=0) x+=t[q],q--;
        u[i]=x%10+'0';
        cc=x/10;
    }
    u[k+1]='\0';
    while (u[0]=='0')
    {
        for (i=0;i<k+1;i++)
            u[i]=u[i+1];
    }
    if (strlen(u)==0)
    {
        u[0]='0',u[1]='\0';
```

```
        }
    }
    int main()
    {
        char s[maxl];
        char t[maxl];
        char u[maxl];
        while (scanf("%s%s",s,t)!=EOF)
        {
            add(s,t,u);
            printf("%s\n",u);
        }
        return 0;
    }
```

二、高精度减法

注：保证大数减小数

```
    #include <stdio.h>
    #include <string.h>
    #include <math.h>
    #define max(a,b) (a>b)?(a):(b)
    #define maxl 2000
    //maxl is the max length of the number
    //should be changed according to the problem
    using namespace std;
    void sub(char *s,char *t,char *u)
    {
        /*pre condition:s and t are two numbers
                        every character in s and t contain only '0'-'9'
                        s and t should be positive numbers
                        s,t satisfy s>t
          post condition:u is s+t
                         make sure u has a length smaller than maxl
                         since s>t, we have u>0
        */
        int i,j;
        int lens=strlen(s),lent=strlen(t);
        int k=max(lens,lent);
        for (i=0;i<lens;i++)
            s[i]-='0';
        for (i=0;i<lent;i++)
            t[i]-='0';
        int cc=0,p=lens-1,q=lent-1;
        int c[maxl];
```

```
    memset(c,0,sizeof(c));
    for (i=k;i>=0;i--)
    {
        int x=-cc;
        if (p>=0) x+=s[p],p--;
        if (q>=0) x-=t[q],q--;
        if (x<0)
        {
            cc=1;
            c[i]=x+10;
        }
        else
        {
            cc=0;
            c[i]=x;
        }
    }
    p=0;
    while (c[p]==0) p++;
    for (i=p,j=0;i<=k;i++,j++)
        u[j]=c[i]+'0';
    u[j]='\0';
    if (strlen(u)==0)
    {
        u[0]='0',u[1]='\0';
    }
}
int main()
{
    char s[maxl];
    char t[maxl];
    char u[maxl];
    while (scanf("%s%s",s,t)!=EOF)
    {
        sub(s,t,u);
        printf("%s\n",u);
    }
    return 0;
}
```

三、高精度乘法

```
#include <stdio.h>
#include <string.h>
#define maxl 2000
#define max(a,b) (a>b)?(a):(b)
//maxl is the max length of the number
//should be changed according to the problem
using namespace std;
void mul(char *s,char *t,char *u)
{
    /*pre condition:s and t are two numbers
                    every character in s and t contain only '0'-'9'
                    s and t should be positive numbers
      post condition:u is s*t
                     make sure u has a length smaller than maxl
    */
    int i,j;
    int lens=strlen(s),lent=strlen(t);
    int k=lens+lent;
    int a[maxl],b[maxl];
    for (i=0;i<lens;i++)
        a[i]=s[i]-'0';
    for (i=0;i<lent;i++)
        b[i]=t[i]-'0';
    int c[maxl];
    memset(c,0,sizeof(c));
    for (i=0;i<lens;i++)
        for (j=0;j<lent;j++)
            c[i+j+2]+=a[i]*b[j];
    int cc=0;
    for (i=k;i>=0;i--)
    {
        int x=cc+c[i];
        c[i]=x%10;
        cc=x/10;
    }
    int p=0;
    while (c[p]==0) p++;
    for (i=0,j=p;j<=k;i++,j++)
        u[i]=c[j]+'0';
    u[i]='\0';
    if (strlen(u)==0)
    {
        u[0]='0',u[1]='\0';
```

```
        }
    }
    int main()
    {
        char s[maxl];
        char t[maxl];
        char u[maxl];
        while (scanf("%s%s",s,t)!=EOF)
        {
            mul(s,t,u);
            printf("%s\n",u);
        }
        return 0;
    }
```

四、高精度除以高精度

注：代码中 div 返回整数部分，mod 返回余数

```
    #include <stdio.h>
    #include <string.h>
    #define maxl 2000
    #define max(a,b) (a>b)?(a):(b)
    //maxl is the max length of the number
    //should be changed according to the problem
    using namespace std;
    void divide(char *s,long long t,char *div,long long *mod)
    {
        /*pre condition:s and t are two numbers
                        every character in s contain only '0'-'9'
                        t should be smaller than 10^18
                        s and t should be positive numbers,t>0
          post condition:div=s/t   mod=s%t
        */
        int i,j;
        long long cc=0;
        int lens=strlen(s);
        int a[maxl],c[maxl];
        for (i=0;i<lens;i++)
            a[i]=s[i]-'0';
        for (i=0;i<lens;i++)
        {
            cc=cc*10+a[i];
            c[i]=cc/t;
            cc=cc%t;
        }
```

```
    int p=0;
    while (c[p]==0) p++;
    for (i=0,j=p;j<lens;i++,j++)
        div[i]=c[j]+'0';
    div[i]='\0';
    if (strlen(div)==0)
    {
        div[0]='0',div[1]='\0';
    }
    *mod=cc;
}
int main()
{
    char s[maxl];
    long long t;
    char div[maxl];
    long long mod;
    while (scanf("%s%lld",s,&t)!=EOF)
    {
        divide(s,t,div,&mod);
        printf("%s %lld\n",div,mod);
    }
    return 0;
}
```

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 <stdio.h>
long long fac_mod(int n,long long m)
{
    long long ans=1;
    int i;
    for (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 (scanf("%lld%lld",&n,&m)!=EOF)
        printf("%lld\n",fac_mod(n,m));
    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 <stdio.h>
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 (scanf("%lld%lld%lld",&a,&b,&c)!=EOF)
    {
        //find (a^b)%c
        printf("%lld\n",power_ntt(a,b,c));
    }
    return 0;
}
```

三、素数

问题 1：如何判断一个数 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 <stdio.h>
#include <math.h>
int is_prime(long long n)
{
    if (n==1) return false;
    long long i;
    for (i=2;i<sqrt(n);i++)
        if (n%i==0) return 0;
    return 1;
}
int main()
{
    long long n;
    while (scanf("%lld",&n)!=EOF)
        printf("%lld\n",is_prime(n));
    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 <stdio.h>
#include <math.h>
long long sum_divisor(long long n)
{
    if (n==1) return 0;
    long long sum=1,i;
    for (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 (scanf("%lld",&n)!=EOF)
        printf("%lld\n",sum_divisor(n));
    return 0;
}
```

四、欧拉筛法：

问题描述：求 1-maxn 中所有的素数

```
#include <stdio.h>
#define maxn 10000+5
int is_prime[maxn];
int get_prime()
{
    int prime[maxn+5];
    int i,t=1;
    for (i=0;i<maxn;i++)
        prime[i]=1;
    is_prime[0]=2;
    for (i=3;i<maxn;i+=2)
    {
        if (prime[i])
        {
            is_prime[t++]=i;
            for (int j=2*i;j<=maxn;j+=i)
                prime[j]=0;
        }
    }
    return t;
}
int main()
{
    int n=get_prime();
    for (int i=0;i<n;i++)
        printf("%d ",is_prime[i]);
    printf("\n");
    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 <stdio.h>
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 (scanf("%lld%lld",&m,&n)!=EOF)
        printf("%lld %lld\n",gcd(m,n),lcm(m,n));
    return 0;
}
```

实训模板库 3——char *类型

一、头文件

```
#include <string.h>
```

二、char *比较

①区分大小写

函数名：strcmp

功能：字符串比较

用法：int strcmp(char *str1, char *str2);

解释：按字典序，str1>str2，返回值>0；两串相等，返回 0；否则返回值<0。

②不区分大小写

函数名：strncmpi

功能：将一个串中的一部分与另一个串比较，不管大小写

用法：int strncmpi(char *str1, char *str2);

三、char *赋值

函数名：strcpy

功能：拷贝一个字符串到另一个

用法：char *strcpy(char *destin, char *source);

解释：执行完后，destin 数组中的值等于 source 中的值

四、char *连接

函数名：strcat

功能：字符串拼接函数

用法：char *strcat(char *destin, char *source);

解释：执行完后，在 destin 字符串后面附加上 source。

五、char *长度

函数名：strlen

功能：字符串长度

用法：size_t strlen(char *s,);

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) (strlen(a)-strlen(b))

六、char *查找函数

```
int strindex(char *s,char *t)
{
    if (!strlen(t)) return -1;  //当 t 为空串时特殊考虑!
    int i,j,k;
    for (i=0;i<strlen(s);i++)
    {
        for (j=i,k=0;s[j] && t[k] && s[j]==t[k];j++,k++);
        if (k==strlen(t)) return i;
    }
    return -1;
}
```

功能：在字符串 s 中查找 t，如果成功找到，返回从左到右第一次找到的位置，否则返回-1。

七、char *读入

①空格为结束标志：scanf("%s",s);，注意 scanf 不读入"\n"和" "

②读入一行数据：fgets(s,maxl,stdin);，注意 fgets 读入"\n"，maxl 为最大长度。

实训模板库 4——数据排序

一、头文件

```
#include <stdlib.h>
```

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

```
#include <stdio.h>
#include <stdlib.h>
#define maxn 10000+5
int cmp(const void *a,const void *b)
{
    long long p=*(long long *)a;
    long long q=*(long long *)b;
    if (p>q) return 1;
    if (p<q) return -1;
    return 0;
}
long long a[maxn];
int main()
{
    int n,i;
    scanf("%d",&n);
    for (i=0;i<n;i++)
        scanf("%d",&a[i]);
    qsort(a,n,sizeof(long long),cmp);
    for (i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\n");
    return 0;
}
```

三、字符串数组排序

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define maxn 1000+5
int cmp(const void *a,const void *b)
{
    char *p=*(char **)a;
    char *q=*(char **)b;
    if (strcmp(p,q)>0) return 1;
    if (strcmp(p,q)<0) return -1;
    return 0;
}
char s[maxn][maxn];
char *ptr[maxn];
int main()
{
    int n,i;
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        scanf("%s",s[i]);
        ptr[i]=s[i];
    }
    qsort(ptr,n,sizeof(char *),cmp);
    for (i=0;i<n;i++)
        printf("%s\n",ptr[i]);
    return 0;
}
```

四、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 <stdio.h>
#include <stdlib.h>
#include <math.h>
#define maxn 10000+5
#define eps 1e-7
struct node
{
    double first,second;
};
typedef struct node node;
int cmp(const void *a,const void *b)
{
    node p=*(node *)a;
    node q=*(node *)b;
    if (fabs(p.first-q.first)>eps)
    {
        if (p.first>q.first) return 1;
        if (p.first<q.first) return -1;
        return 0;
    }
    if (p.second>q.second) return -1;
    if (p.second<q.second) return 1;
    return 0;
}
node a[maxn];
int main()
{
    int n,i;
    scanf("%d",&n);
    for (i=0;i<n;i++)
        scanf("%lf%lf",&a[i].first,&a[i].second);
    qsort(a,n,sizeof(node),cmp);
    for (i=0;i<n;i++)
        printf("%lf %lf\n",a[i].first,a[i].second);
    return 0;
}
```

五、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.    typedef struct node node;
6.    int cmp(const void *a,const void *b)
7.    {
8.        node p=*(node *)a;
9.        node q=*(node *)b;
10.       if (p.cnt!=q.cnt)  return q.cnt-p.cnt;
11.       return p.n-q.n;
12.   }
```

note：代码中 n 表示数据大小，cnt 表示数据出现次数。

note：由于 n 和 cnt 是 int 类型，两个 int 类型做减法不会越界（返回值也是 int 类型），如果是 long long 或者 double 只能按照上面的模板写。

（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：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。

②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，解决方法：while (scanf(“%d”,&n)!=EOF)。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）