1.三维坐标结构体数组

> 请定义一个表示三维坐标的结构体类型，用于存储点的三维坐标整数值。
> 在主函数中，输入N个点的三维坐标(X, Y, Z)，并存储在结构体数组中。请你找出它们之中z轴坐标最大的点，并输出该点的三维坐标值。为简化问题，假定z轴坐标最大的点是唯一的。
>
> ---
>
> 输入
>
> > 输入包含$N+1$行：
> > 第一行是正整数$N(0<N<1000)$。
> > 第二行到第$N+1$行，每行都有三个整数，是某个点的三维坐标值X Y Z。邻近两数用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出z轴坐标最大的点的三维坐标值。邻近两数用一个空格隔开。

```c
#include <stdio.h>
typedef struct {
    int x;
    int y;
    int z;
} Point;

int main() {
    int n,i;
    int max_index = 0;
    Point points[1000];
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
```

```
        scanf("%d %d %d", &points[i].x, &points[i].y,
&points[i].z);
    }
    for (i = 1; i < n; i++) {
        if (points[i].z > points[max_index].z) {
            max_index = i;
        }
    }
    printf("%d %d %d\n", points[max_index].x,
points[max_index].y, points[max_index].z);
}
```

2.初识结构体之座机电话号码

> 请定义一个表示电话号码的结构体类型。
> 电话号码包含区号（最多4位）和区内电话号码（最多8位）。在一个区内的电话号码之间互相拨号时，不拨区号，否则必须先拨区号。
> 在主函数中，输入任意两个电话号码A和B（区号和区内号码之间使用一个空格隔开），输出A给B打电话时拨的号码。
>
> ---
>
> 输入
>
> > 输入包含两行，分别是电话号码A和B。区号和区内号码之间使用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出A给B打电话时拨的号码。

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

```c
typedef struct {
    char area[5];
    char number[9];
} Phone;

int main() {
    Phone a, b;
    scanf("%s %s", a.area, a.number);
    scanf("%s %s", b.area, b.number);
    if (strcmp(a.area, b.area) == 0) {
        printf("%s", b.number);
    } else {
        printf("%s%s", b.area, b.number);
    }
}
```

3.初识结构体之局域网判断

> 互联网上IP地址的表示方式为：x.y.z.m，其中x、y、z和m都是正整数。可以通过IP地址来区分同一局域网中的各个计算机。
> 请定义一个表示IP地址的结构体类型，其中含有四个成员，分别为四个int类型的整数。然后输入两个IP地址，我们根据前两个成员值是否相同判断它们是否处于同一个局域网中，根据判断，输出“TRUE”或者“FALSE”(不输出引号)。
>
> ---
>
> 输入
>
> > 输入包含两行，分别是两个IP地址。
>
> ---
>
> 输出

> 依据题意，输出“TRUE”或者“FALSE”(不输出引号)。

```
#include <stdio.h>
typedef struct {
    int a;
    int b;
    int c;
    int d;
}IP;
int main() {
    IP ip[2];
    scanf("%d.%d.%d.%d", &ip[0].a, &ip[0].b, &ip[0].c,
&ip[0].d);
    scanf("%d.%d.%d.%d", &ip[1].a, &ip[1].b, &ip[1].c,
&ip[1].d);
    if (ip[0].a == ip[1].a && ip[0].b == ip[1].b) {
        printf("TRUE");
    }
    else {
        printf("FALSE");
    }
}
```

4.初识结构体之日期函数

> 请定义一个表示日期的结构体类型，用于存储年、月、日三个成员。
> 在主函数中，输入两个日期。请你为上述结构体定义相关函数并调用，
> 找出两个日期之间间隔的天数。

输入

> 输入包含两行：
> 第一行是第一个日期的年、月、日，邻近数据之间用一个空格隔开。

> 第二行是第二个日期的年、月、日，邻近数据之间用一个空格隔开。

输出

> 输出两个日期间隔的天数。

```
#include <stdio.h>
typedef struct {
    int year;
    int month;
    int day;
} Date;

int is_leap(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400
== 0);
}
int get_month_days(int year, int month) {
    int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31};
    if (month == 2 && is_leap(year)) return 29;
    return days[month - 1];
}
int total_days(Date d) {
    int total = 0,y,m;
    for (y = 1; y < d.year; y++) {
        total += is_leap(y) ? 366 : 365;
    }
    for (m = 1; m < d.month; m++) {
        total += get_month_days(d.year, m);
    }
    total += d.day;
    return total;
}
int main() {
    int num1, num2,days;
    Date date[2];
```

```
    scanf("%d %d %d", &date[0].year, &date[0].month,
&date[0].day);
    scanf("%d %d %d", &date[1].year, &date[1].month,
&date[1].day);
    num1 = date[0].year * 10000 + date[0].month * 100 +
date[0].day;
    num2 = date[1].year * 10000 + date[1].month * 100 +
date[1].day;
    if (num1 > num2) {
        Date temp = date[0];
        date[0] = date[1];
        date[1] = temp;
    }
    days = total_days(date[1]) - total_days(date[0]);
    printf("%d", days);
}
```

5.地球的人造卫星远近问题

> 地球的人造卫星如果按它们在轨道上的功能来进行分类，可分为观测站、中继站、基准站和轨道武器四类。人造卫星的运动服从开普勒行星运动定律，其轨道一般是以地心为焦点的椭圆，特殊情况下是以地心为中心的圆。
> 下面这五个开普勒轨道常数是描述某颗人造地球卫星的重要数据：
> 1）轨道倾角i，是赤道平面与卫星轨道平面间的夹角。
> 2）升交点赤经Ω，是从春分点到卫星升交点的经度。
> 3）近地点幅角ω，是地心与升交点连线和地心与近地点连线间的夹角。
> 4）椭圆半长轴a。
> 5）椭圆偏心率e。
> 请定义一个表示人造卫星轨道参数的结构体类型，用于存储上述五项主要参数。
> 在主函数中输入N个卫星的参数（五个参数均为float范围内的数据），存储在结构体数组中。请你找出N个卫星之中离地球最近（即椭圆半长轴最小）的卫星，并输出该卫星的各项数据；以及离地球最远（即椭圆半

长轴最大）的卫星的各项数据。
为简化问题，假定距离地球最近的卫星只有一个，距离地球最远的卫星也只有一个。

输入

> 输入包含N + 1行：
> 第一行是正整数N（1 < N < 20）。
> 第二行到第N + 1行，每行都有五个float范围内的浮点数，是某个卫星的轨道倾角、升交点赤经、近地点幅角、椭圆半长轴和椭圆偏心率。邻近两数用一个空格隔开。

输出

> 输出两行，分别是离地球最近的卫星轨道参数、离地球最远的卫星轨道参数。邻近两数用一个空格隔开。小数点后必须保留2位有效数字（四舍五入），不足补零。

```
#include <stdio.h>
typedef struct {
    float i;
    float Omega;
    float omega;
    float a;
    float epsilon;
}Ellipses;
int main() {
    int n,j,max,max_index=0,min,min_index=0;
    Ellipses ellipses[20];
    scanf("%d", &n);
    for(j=0;j<n;j++){
        scanf("%f %f %f %f %f", &ellipses[j].i,
&ellipses[j].Omega, &ellipses[j].omega, &ellipses[j].a,
&ellipses[j].epsilon);
```

```
            if (j==0) {
                max = ellipses[j].a;
                min = ellipses[j].a;
            }
            if (max < ellipses[j].a) {
                max = ellipses[j].a;
                max_index = j;
            }
            if (min > ellipses[j].a) {
                min = ellipses[j].a;
                min_index = j;
            }
        }
        printf("%.2f %.2f %.2f %.2f %.2f\n",
ellipses[min_index].i, ellipses[min_index].Omega,
ellipses[min_index].omega, ellipses[min_index].a,
ellipses[min_index].epsilon);
        printf("%.2f %.2f %.2f %.2f %.2f",
ellipses[max_index].i, ellipses[max_index].Omega,
ellipses[max_index].omega, ellipses[max_index].a,
ellipses[max_index].epsilon);
}
```

6.挑选面积最大的卡片

> 乐乐在幼儿园认识了长方形，回到家里立刻翻出红色、绿色、黄色……等不同颜色的N张卡纸，用每张卡纸分别剪出了一个端端正正的长方形卡片。这N个长方形长短不一。乐乐想在其中一个卡片上粘贴老师奖励的小红花，可是乐乐不知道哪张长方形卡片面积最大，可以粘贴所有的小红花。
> 请你定义卡片的结构体类型，内含三个数据成员：颜色、长、宽。
> 并将输入的N个卡片信息存入结构体数组中。然后从这N张长方形卡片中挑选出面积最大的卡片，输出其颜色信息。
> 假定：面积最大的卡片只有一张。

输入

> 输入包含N + 1行：
> 第一行是正整数N。
> 第二行到第N + 1行，每行都有如下三项，邻近两项之间用一个空格隔开。
> 1）一个不含空白符的卡片颜色字符串（不超过9个字符）。
> 2）一个int范围内的正整数（卡片长度）。
> 3）一个int范围内的正整数（卡片宽度）。

输出

> 输出面积最大的卡片的颜色。

```c
#include <stdio.h>
typedef struct{
    char color[10];
    int x;
    int y;
}Color;
int main() {
    int n,i,squre,max_squre=0,max_index=0;
    Color color[20];
    scanf("%d", &n);
    for(i=0;i<n;i++){
        scanf("%s %d %d", color[i].color, &color[i].x,
&color[i].y);
        squre = color[i].x * color[i].y;
        if (squre > max_squre) {
            max_squre = squre;
            max_index = i;
        }
    }
    printf("%s", color[max_index].color);
}
```

7.简单的月度排行榜问题

> 大名鼎鼎的某文学城网站系统要更新升级。假定所有书的关联数据为下面这几项：
> 1）书名，不含空白符的字符串，不超过59个字符。
> 2）作者，不含空白符的字符串，不超过19个字符。
> 3）本月月票数，整型数。
> 4）本月点击数，整型数。
> 5）完成字数，整型数。
> 请你定义书的结构体类型，内含上述数据成员。
> 然后给该文学城网站编写月度月票排行榜的程序，根据本月月票数从高到低排序（建议利用指针数组进行排序），最后输出排序后的书名。
> 为了简单起见，假定N本书的月票各不相同。

输入

> 输入包含N + 1行：
> 第一行是正整数N。
> 第二行到第N + 1行，每行都有五个数据，邻近两数用一个空格隔开。这五个数据分别是某本书的书名、作者、本月月票数、本月点击数和完成字数。

输出

> 输出N行，分别是根据本月月票数从高到低排序后的书名。

```c
#include <stdio.h>
typedef struct {
    char name[60];
    char author[20];
    int vips;
```

```
    int clicks;
    int words;
}Book;
int main() {
    int n,i,j;
    Book book[20],*p,temp;
    scanf("%d", &n);
    for(p=book;p<book+n;p++){
        scanf("%s %s %d %d %d", p->name, p->author, &p-
>vips, &p->clicks, &p->words);
    }
    //这里运用冒泡排序法进行排序，按照vips的数量进行排序，用别的排序
方法也可以
    for(i = 0; i < n-1; i++) {
        for(j = 0; j < n-i-1; j++) {
            if(book[j].vips < book[j+1].vips) {
                temp = book[j];
                book[j] = book[j+1];
                book[j+1] = temp;
            }
        }
    }
    for(p=book;p<book+n;p++){
        printf("%s\n", p->name);
    }
}
```

8.地球的远亲与近邻

> 在银河系中，有着诸多的恒星陪伴地球，其中就有大名鼎鼎的太阳。丁丁历险归来，对恒星产生了巨大的兴趣，他要找出距离地球最近的恒星和最远的恒星，请你帮助他完成这个愿望吧！为了简单起见，采用空间直角坐标系，且地球位于坐标系原点。
> 请定义恰当的恒星结构体类型，输入N个恒星的名称和三维坐标，存储在结构体数组中。输出距离地球最近的恒星名称和最远的恒星名称。假定距离地球最近的恒星只有一个，距离地球最远的恒星也只有一个。

输入

> 输入包含N + 1行：
> 第一行是正整数N。
> 第二行到第N + 1行，每行都有四个数据，邻近两项用一个空格隔开。
> 这四个数据分别是某恒星的名称（不含空白符的字符串，字符数不超过59个）、恒星的三维坐标（int范围内的整数）。

输出

> 输出两行，分别是距离地球最近的恒星名称和最远的恒星名称。

```c
#include <stdio.h>
#include <math.h>
typedef struct {
    char name[60];
    int x;
    int y;
    int z;
}Star;
int main() {
    int n,max_index,min_index;
    float distance,max_disrance,min_distance;
    Star star[20],*p,temp;
    scanf("%d", &n);
    for(p=star;p<star+n;p++){
        scanf("%s %d %d %d", p->name, &p->x, &p->y, &p->z);
        distance = sqrt(pow(p->x, 2) + pow(p->y, 2) + pow(p->z, 2));
        if(p==star){
            max_disrance = distance;
            min_distance = distance;
            max_index = 0;
            min_index = 0;
```

```
        }
        if (distance > max_disrance) {
            max_disrance = distance;
            max_index = p-star;
        }
        if (distance < min_distance) {
            min_distance = distance;
            min_index = p-star;
        }
    }
    printf("%s\n", star[min_index].name);
    printf("%s", star[max_index].name);
}
```

9.有理数运算函数

> 有理数可以表示为两个整数的商。例如1.5是一个有理数，表示为3/2。在计算时，使用有理数比使用浮点数更有优势，有理数是精确的，浮点数则不是。
> 因此，请你设计一个表示有理数的数据类型(假定整数用int类型即可)，然后设计一个计算有理数之和的函数、一个计算有理数之积的函数。这两个函数的返回值均为有理数。
> 在主函数中输入2个有理数，分别调用两个函数计算它们的和以及它们的乘积，最后输出这两个函数的返回值。
> 注意：有理数运算函数的计算结果应该化简到最简形式。
>
> ---
>
> 输入
>
> > 输入两项，分别是两个有理数。两个有理数之间用一个空格隔开。
>
> ---
>
> 输出

> 输出两个有理数的和以及它们的乘积。两个结果之间用一个空格隔开。

```
#include <stdio.h>
typedef struct {
    int numerator;
    int denominator;
} Fraction;

int FIND_GCD(int a, int b) {
    a = a>0?a:-a;
    b = b>0?b:-b;
    if (b == 0) return a;
    return FIND_GCD(b, a % b);
}

void simplify(Fraction *f) {
    int gcd = FIND_GCD(f->numerator, f->denominator);
    f->numerator /= gcd;
    f->denominator /= gcd;
    if (f->denominator < 0) {
        f->numerator *= -1;
        f->denominator *= -1;
    }
}

Fraction add(Fraction a, Fraction b) {
    Fraction result;
    int lcm = a.denominator * b.denominator /
FIND_GCD(a.denominator, b.denominator);

    result.numerator = a.numerator * (lcm / a.denominator) +
b.numerator * (lcm / b.denominator);
    result.denominator = lcm;
    simplify(&result);
    return result;
}

Fraction multiply(Fraction a, Fraction b) {
```

```
    Fraction result;
    result.numerator = a.numerator * b.numerator;
    result.denominator = a.denominator * b.denominator;
    simplify(&result);
    return result;
}

int main() {
    Fraction frac[2],sum,product;
    int i;
    for (i = 0; i < 2; i++) {
        scanf("%d/%d", &frac[i].numerator,
&frac[i].denominator);
    }
    sum = add(frac[0], frac[1]);
    product = multiply(frac[0], frac[1]);
    printf("%d/%d %d/%d\n",
           sum.numerator, sum.denominator,
           product.numerator, product.denominator);
}
```

10.地精马普的货物数量统计

> 地精马普开了一个魔法用品店，出售的物品种类丰富。可是众多的魔法用品及其价格充斥在马普的脑袋中，可怜的马普经常忙中出错，不是卖便宜了就是卖贵了，因此老顾客们和附近其它地精店主总来抱怨马普的错误。马普听了吟游诗人说起冰雪大陆上流行的电脑后，认为魔力非凡的电脑一定能帮助他。可以把魔法用品店里的所有商品、价格以及库存量都存在一台电脑中。
> 他请你来设计一种数据类型，可以存储商品名（不含空白符的字符串，不超过59个字符），价格（int范围内的整数）以及其库存（int范围内的正整数）。
> 在主函数中定义结构体数组，存储N种货物的信息，并且编写程序帮助马普统计所有货物的数量之和。

输入

> 输入包含N + 1行：
> 第一行是正整数N。
> 第二行到第N + 1行，每行都有三个数据，邻近两项用一个空格隔开。
> 这三个数据分别是商品名（不含空白符的字符串，不超过59个字符），价格（int范围内的整数）以及其库存（int范围内的正整数）。

输出

> 输出所有货物的数量之和。

```c
#include <stdio.h>
typedef struct {
    char name[60];
    int value;
    int num;
} Goods;
int main() {
    int n, i, sum = 0;
    Goods goods[20], temp;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%s %d %d", goods[i].name, &goods[i].value,
&goods[i].num);
        sum+=goods[i].num;
    }
    printf("%d", sum);
}
```