写在最前的tips(常见错误若干)

- **数组大小不要用变量** 很多同学定义数组大小用了变量，虽然C99支持变长数组（VLA），但强烈不推荐
- **关于多组数据** 不明白多组数据的含义，不会多组数据的读入，输出没有换行的意识，以为非要一次性读完所有数据再一起输出
- **不会用Ctrl + Z模拟EOF**
- **缓冲区的刷新，读字符时回车和空格不会处理**
- **对类型不够敏感，不会强制类型转换，%lld对应long long的输入和输出，%d对应int的输入和输出**

A. HugeGun学姐做除法

解题思路

本题考查点为浮点数的输入输出和运算。

根据 $hint$ ，想要除法的结果是实数，必须使用 $double$ 类型的运算，因此定义a,b,c为 $double$ 类型。根据除法运算性质，本题共有两种运算顺序，即 $(a \div b) \div c$ 和 $a \div (b \div c)$ ，分别求出来然后取平均值即可。

注意：标准 C 规定$double$类型的输入使用 $\%lf$ 控制，输出使用 $\%f$ 控制，输出写成 $\%lf$ 虽然可以执行，但是是不标准的。

另：使用%d读入的同学，可以使用类似“ans1=1.0*a/b/c,ans2=a/(1.0*b/c)”的方式进行隐式类型转换，或者使用“ans1=(double)a/b/c,ans2=a/((double)b/c)”进行强制类型转换。请同学们思考：为什么要把乘号或(double)放在这里？放在别的地方可以吗？可以自己动手实践一下验证自己的猜测。

AC代码

```c
#include <stdio.h>

int main() {
    double a, b, c;
    double ans1, ans2, ans;
    scanf("%lf%lf%lf", &a, &b, &c);
    ans1 = a / b / c;
    ans2 = a / (b / c);
    ans = (ans1 + ans2) / 2;
    printf("%.7f", ans);
    return 0;
}
```

B. 粉红兔装lyd

解题思路

本题考查的是经典的鸡兔同笼问题。先假设兔笼里全都是兔子，则应该有$4h$只腿，比实际情况多了$4h-f$只腿，再除以2即可得lyd的数量$2h-f/2$。最后$h-(2h-f/2)=f/2-h$为兔子的数量。

AC代码

```c
#include <stdio.h>

int main() {
    int h, f;
    scanf("%d%d", &h, &f);
    printf("%d %d", 2 * h - f / 2, f / 2 - h);
    return 0;
}
```

C. 认真+耐心=AC!

解题思路

根据题目中的Hint，这道题可以通过控制 *scanf* 读入格式来分别读入整数和小数部分，再分别求和

AC代码

```
#include<stdio.h>
int main(){
    long long a,a0,b,b0;
    int T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%lld.%lld %lld.%lld",&a,&a0,&b,&b0);
        printf("%lld.%lld\n",a+b,a0+b0);
    }
    return 0;
}
```

D. 火仙花数

解题思路

根据题目的定义，火仙花数是一个四位数，而四位数的个数很少，我们可以考虑从小到大枚举所有的四位数，然后检查这个数是否是火仙花数。

那么我们怎么找到第n个火仙花数呢，可以用一个变量$count$来记一下这是第几个火仙花数，如果$count = n$那么说明这个数是第n个火仙花数。

AC代码

```
#include<stdio.h>
int main()
{
    int n, i, a, b, c, d, count = 0;
    scanf("%d", &n);
    for (i = 1000; i <= 9999; ++i) {
        a = i % 10;          //个位
        b = (i / 10) % 10;   //十位
        c = (i / 100) % 10; //百位
        d = (i / 1000);      //千位
        if(a * a * a * a + b * b * b * b + c * c * c * c + d * d * d * d == i){
            ++count;
            if (count == n)
                printf("%d", i);
        }
    }
    return 0;
}
```

但如果你输出所有的火仙花数，你会发现它只有三个，那么我们可以做一个if怪：

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    if (n == 1)
        printf("1634");
    else if (n == 2)
        printf("8208");
    else
        printf("9474");
    return 0;
}
```

或者使用出题人的写法：

```c
#include<stdio.h>
int ans[4] = { 0, 1634, 8208, 9474 };
int main()
{
    int n;
    scanf("%d", &n);
    printf("%d", ans[n])
}
```

E. ASCII

解题思路

本题考察点为$ASCII$码和字符的转换。需要注意的点如下：

1. 使用$\%c$输出数字即可输出$ASCII$码值为该数字的字符。
2. 使用$\%d$输出字符即可输出字符对应的$ASCII$码值。
3. 求两个数的最大值或最小值可使用三目运算符简化代码。比如 $a > b\ ?\ a : b$ 在 $a > b$ 时返回值是 a，在 $a < b$ 时返回值是 b。
4. 使用 `scanf("%c %c", &c, &d);` 读取字符时，c 会读到前面一行的换行符，应该在第一个 $\%c$ 之前加个空格吃掉前面的空白字符。

AC代码

```
#include <stdio.h>

int main() {
    int tp, a, b;
    char c, d;
    while (scanf("%d", &tp) != EOF) {
        if (tp == 1) {
            scanf("%d%d", &a, &b);
            printf("%c\n", a > b ? a : b);
        }
        else {
            scanf(" %c %c", &c, &d);  //用空格吃掉前面的换行符
            printf("%d\n", c < d ? c : d);
        }
    }
    return 0;
}
```

F. 商的分离

解题思路

这道题进行除法计算的时候，由于小数部分以浮点数保存，可能出现-0.00000的情况，因此需要使用eps消除-0的情况。当然，在 *math.h* 库中提供了 *ceil()* 和 *floor()* 两个函数，分别用来向上取整和向下取整，可以使用这两个函数处理整数部分后，先消除小数部分-0的情况，再将整数部分处理成 *int* 类型

AC代码

使用eps处理小数

```
#include<stdio.h>
#include<math.h>
#define eps 1e-12
int main()
{
    double u,v,a,b;
    int result;
    while(scanf("%lf %lf",&u,&v)!=EOF)
    {
        a=u/v;
        result=(int)a;
        b=(u/v)-result;
        printf("%d %.9lf\n",result,b+eps);
    }
    return 0;
}
```

使用 ceil() 和 floor() 函数

```
#include<stdio.h>
#include<math.h>
int main()
{
    double u,v,a,b;
    int result;
    while(scanf("%lf %lf",&u,&v)!=EOF)
    {
        if(u*v<0)
        {
            a=ceil(u/v);
            b=(u/v)-a;
        }
        else
        {
            a=floor(u/v);
            b=(u/v)-a;
        }
        result=(int)a;
        printf("%d %.9lf\n",result,b);
    }
```

```
    return 0;
}
```

G. 摸鱼大赛

解题思路

本题是ppt中c2.9的加强版，主要体现在数据规模大了很多，所以不可能给每个数据命名一个变量，最好的解决方案就是用一个**数组**来存储每次读入的数据，再通过**循环**逐个比较，同时统计蒙对的个数，最后再根据蒙对的个数输出'*'或者"Your prediction is terrible!"即可.

注意，数组大小尽量不要开恰好所需个数，如：定义为a[1000]的数组只存在a[0],a[1]...a[999]，在访问a[1000]时会造成数组越界问题；字符串在最后也会加入'\0'字符。故可以将数组大小稍稍开大一点，如本题n的数据范围最大到1000，数组大小就可以开到1005,1010等稍大的数以避免某些错误。

AC代码

```
#include<stdio.h>
int n, m, rk[1010], pre[1010];

int main()
{
    int i;
    scanf("%d%d", &n, &m);
    for (i = 1; i <= n; ++i)
        scanf("%d", &rk[i]);
    while (m--) {
        int bingo = 0;
        for (i = 1; i <= n; ++i)
            scanf("%d", &pre[i]);
        for (i = 1; i <= n; ++i)
            if (pre[i] == rk[i])
                ++bingo;
        if (!bingo)
            printf("Your prediction is terrible!");
        for (i = 1; i <= bingo; ++i)
            printf("*");
        printf("\n");
    }
    return 0;
}
```

H. 直线与圆

解题思路

本题的核心其实是*两点式的直线方程*以及*点到直线的距离公式*这两个**数学知识点**，即给定两点求出对应的$Ax+By+C=0$中的系数A,B,C以及根据A,B,C求出距离$d=\frac{|Ax+By+C|}{\sqrt{A^2+B^2}}$，再根据$d$与$r$的关系得到结果即可。

另一点需要注意的是本题对精度的要求，众所周知math.h里的函数都是有精度误差的(雾)，因此浮点数的严格相等往往难以满足，所以需要用一个极小数eps进行判断，只要d与r之差的绝对值小于eps，就可以认为它们相等(有点类似数分中的极限思想)

AC代码

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

int main() {
    double x1, y1, x2, y2, x3, y3, r, a, b, c, d;
    while(scanf("%lf%lf%lf%lf%lf%lf%lf",&x1,&y1,&x2,&y2,&x3,&y3,&r)!=EOF){
        if (fabs(x1 - x2) <= 1e-12) {
            a = 1;
            b = 0;
            c = -x1;
        }
        else {
            a = (y2 - y1) / (x2 - x1);
            b = -1;
            c = y1 - a * x1;
        }
        d = fabs(a * x3 + b * y3 + c) / sqrt(a * a + b * b);
        if (fabs(d - r) <= 1e-12) {
            printf("1\n");
        }
        else if (d < r) {
            printf("2\n");
        }
        else {
            printf("0\n");
        }
    }
    return 0;
}
```

I. Chem is try

解题思路

有时根据题目中所给的信息，我们实际上可以少考虑一些特殊情况，从而简化自己的代码，加快速度(**所以说审题真的很重要**).比如就以本题为例，既然题目已明确指出分子质量肯定不会小于1，所以最后格式化输出时就只需要考虑'+'的情况.

此外，如果有同学觉得手动判断一个字符是否是数字或者字母比较麻烦，不妨研究一下<ctype.h>中的函数，或许可以提高你代码的易读性(AC代码中就使用到了其中的2个)

本题的另外一个坑点就是最开始读入元素时,如果使用char类型读取，由于'\n',' '都是char类型，所以可能会在循环中出现问题.(不过这个问题相信同学们可以通过调试发现.)这里采取的方案是直接读入一个字符串，把换行符、空格等全部读进来，避免不必要的是非.

AC代码

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

double m[100];
char s[100];

int main()
{
    int n,t;
    char ele[10]={0};
    double wei;
    scanf("%d",&n);
    //printf("n=%d\n",n);
    for (int i=0; i<n; i++)
    {
        scanf("%s",ele);
        scanf("%lf",&wei);
        //printf("ele=%c,wei=%f\n",ele[0],wei);
        m[ele[0]-'A'] = wei; //将分子量赋值给数组中的对应元素
    }
    scanf("%d",&t);
    //printf("t=%d\n",t);
    double tot = 0;
    int exp = 0;
    while(t--)
    {
        tot = 0;
        exp = 0;
        scanf("%s",s);
        for (int i=0; s[i]; i++)
        {
            if (isupper(s[i]))
            {
                if(isdigit(s[i+1]))
                {
                    tot += (s[i+1]-'0')*m[s[i]-'A'];
                }
```

```
                else
                    tot += m[s[i]-'A'];
            }
        }
        printf("M(%s) = ",s);
        while(tot >= 10)
        {
            tot /= 10;
            exp += 1;
        }
        printf("%.5fE+%03d\n",tot,exp);
    }
}
```

J. 交作业啦

解题思路

本题根据题目要求，只需要分别统计 *Corpse* 提交作业时是提前交还是迟交即可，但需注意进行除法运算时的变量类型。

AC代码

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int i,temp;
    int ontime=0,delay=0;
    int ontime_count=0,delay_count=0;
    for(i=0;i<n;i++)
    {
        scanf("%d",&temp);
        if(temp>=0)
        {
            ontime++;
            ontime_count+=temp;
        }
        else
        {
            delay++;
            delay_count+=temp;
        }
    }
    ontime=(ontime*100)/n;
    if(ontime>=90)
    {
        printf("%.2lf",(double)(ontime_count+delay_count)/(double)(n));
    }
    else if(ontime>=50)
    {
        printf("%d%%",ontime);
    }
    else
    {
        printf("My, please help Corpse!\n");
        printf("%d%%\n%.2lf",(delay*100)/n,delay_count/(double)delay);
    }
    return 0;
}
```

K. Potassium的成绩单

解题思路

这道压轴题……首先要先判断代表两只手的矩形和成绩单是否有交集，然后在成绩单总面积上扣除被遮住的面积，但需注意有可能代表两只手的矩形之间有公共部分，要避免重复扣除。

出题人的hint：本题考查读题能力，对角线不代表左下右上，为了不让大家掉进这个坑，样例还专门给了一组有关数据，可还是比较遗憾没有人能在场上做出来...同学们以后读题一定要仔细再仔细！加油！

AC代码

出题人的标程

```
#include<stdio.h>
int min(int a,int b){return a>b?b:a;}
int max(int a,int b){return a>b?a:b;}
int main(){
   int n,x1,x2,x3,x4,x5,x6,y1,y2,y3,y4,y5,y6;
   scanf("%d",&n);
   while(n--){
      int a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6;

 scanf("%d%d%d%d%d%d%d%d%d%d%d%d",&a1,&b1,&a2,&b2,&a3,&b3,&a4,&b4,&a5,&b5,&a6,&b
6);
      x1=min(a1,a2);x2=max(a1,a2);y1=min(b1,b2);y2=max(b1,b2);
      x3=min(a3,a4);x4=max(a3,a4);y3=min(b3,b4);y4=max(b3,b4);
      x5=min(a5,a6);x6=max(a5,a6);y5=min(b5,b6);y6=max(b5,b6);
      long long S=1LL*(x2-x1)*(y2-y1);
      if(min(x2,x4)>max(x1,x3)&&min(y2,y4)>max(y1,y3))
            S-=1LL*(min(x2,x4)-max(x1,x3))*(min(y2,y4)-max(y1,y3));
      if(min(x2,x6)>max(x1,x5)&&min(y2,y6)>max(y1,y5))
            S-=1LL*(min(x2,x6)-max(x1,x5))*(min(y2,y6)-max(y1,y5));
      if(min(x4,x6)>max(x3,x5)&&min(y4,y6)>max(y3,y5)){
         int s2=min(x4,x6),s1=max(x3,x5),t2=min(y4,y6),t1=max(y3,y5);
         if(min(x2,s2)>max(x1,s1)&&min(y2,t2)>max(y1,t1))
               S+=1LL*(min(x2,s2)-max(x1,s1))*(min(y2,t2)-max(y1,t1));
      }
      printf("%lld\n",S);
   }
   return 0;
}
```

这份代码用了指针。。。

```
#include<stdio.h>
#include<math.h>
void change(long long *a,long long *b)//交换数字顺序保证左下到右上对角线
{//此处使用了自定义函数和指针相关知识，仅供参考
    long long t;
    if(*a>*b)
    {
        t=*a;
```

```
        *a=*b;
        *b=t;
    }
}
void calculated_area(long long x1,long long y1,long long x2,long long y2,long
long x3,long long y3,long long x4,long long y4,long long *return_x1,long long
*return_y1,long long *return_x2,long long *return_y2)//返回公共区域对角线两端点的坐标
{
    if(x3<x1&&x4<x2)
    {
        *return_x2=x4;
        *return_x1=x1;
    }
    else if(x3>x1&&x4>x2)
    {
        *return_x2=x2;
        *return_x1=x3;
    }
    else if(x3>=x1&&x4<=x2)
    {
        *return_x2=x4;
        *return_x1=x3;
    }
    else if(x3<=x1&&x4>=x2)
    {
        *return_x2=x2;
        *return_x1=x1;
    }
    if(y4>y2&&y3>y1)
    {
        *return_y2=y2;
        *return_y1=y3;
    }
    else if(y4<y2&&y3<y1)
    {
        *return_y2=y4;
        *return_y1=y1;
    }
    else if(y4<=y2&&y3>=y1)
    {
        *return_y2=y4;
        *return_y1=y3;
    }
    else if(y4>=y2&&y3<=y1)
    {
        *return_y2=y2;
        *return_y1=y1;
    }
}
int main()
{
    long long x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6;
    int n;
    scanf("%d",&n);
    while(n--)
    {
        //读入
```

```
        scanf("%lld %lld %lld %lld %lld %lld %lld %lld %lld %lld %lld
%lld",&x1,&y1,&x2,&y2,&x3,&y3,&x4,&y4,&x5,&y5,&x6,&y6);
        change(&x1,&x2);
        change(&y1,&y2);
        change(&x3,&x4);
        change(&y3,&y4);
        change(&x5,&x6);
        change(&y5,&y6);
        long long s=(y2-y1)*(x2-x1);//计算白色总面积
        long long temp_x1,temp_x2,temp_y1,temp_y2;
        //判断 1号矩形是否有遮挡
        if(x3<x2&&x4>x1&&y3<y2&&y4>y1)
        {

calculated_area(x1,y1,x2,y2,x3,y3,x4,y4,&temp_x1,&temp_y1,&temp_x2,&temp_y2);
            s-=(temp_x2-temp_x1)*(temp_y2-temp_y1);
        }
        //判断 2号矩形是否有遮挡
        if(x5<x2&&x6>x1&&y5<y2&&y6>y1)
        {

calculated_area(x1,y1,x2,y2,x5,y5,x6,y6,&temp_x1,&temp_y1,&temp_x2,&temp_y2);
            s-=(temp_x2-temp_x1)*(temp_y2-temp_y1);
        }
        //判断 1号矩形和 2号矩形是否有遮挡
        if(x3<x6&&x4>x5&&y3<y6&&y4>y5)
        {

calculated_area(x3,y3,x4,y4,x5,y5,x6,y6,&temp_x1,&temp_y1,&temp_x2,&temp_y2);
            //判断 1号矩形和 2号矩形重叠部分和原矩形是否有重叠
            if(temp_x1<x2&&temp_x2>x1&&temp_y1<y2&&temp_y2>y1)
            {

calculated_area(x1,y1,x2,y2,temp_x1,temp_y1,temp_x2,temp_y2,&temp_x1,&temp_y1,&t
emp_x2,&temp_y2);
                s+=(temp_x2-temp_x1)*(temp_y2-temp_y1);
            }
        }
        //输出
        printf("%lld\n",s);
    }
    return 0;
}
```

另一份代码

```
#include<stdio.h>
#define LL long long
LL ans;
int x[10], y[10];

int min(int a, int b) { return a < b ? a : b; }
int max(int a, int b) { return a > b ? a : b;  }

int cal(int la, int ra, int lb, int rb) {
    return max(0, min(ra, rb) - max(la, lb));
```

```
}
//两个矩形的面积交等于 长的交 乘 宽的交
LL intersec(int a, int b) {
    int secx = cal(x[a], x[a + 1], x[b], x[b + 1]);
    int secy = cal(y[a], y[a + 1], y[b], y[b + 1]);
    return 1ll * secx * secy;
}

int main()
{
    int t, i, a, b;
    scanf("%d", &t);
    while (t--) {
        for (i = 1; i <= 6; ++i)
            scanf("%d%d", &x[i], &y[i]);
        //限定对角线为从左下到右上
        for (i = 1; i <= 5; i += 2) {
            a = x[i]; b = x[i + 1];
            x[i] = min(a, b); x[i + 1] = max(a, b);
            a = y[i]; b = y[i + 1];
            y[i] = min(a, b); y[i + 1] = max(a, b);
        }
        ans = 1ll * (x[2] - x[1]) * (y[2] - y[1]);
        //一只手和试卷的交
        ans -= intersec(1, 3);
        ans -= intersec(1, 5);
        //将两只手的矩形求交
        x[3] = max(x[3], x[5]); x[4] = min(x[4], x[6]);
        y[3] = max(y[3], y[5]); y[4] = min(y[4], y[6]);
        //交后的矩形与试卷求交
        if (x[3] <= x[4] && y[3] <= y[4])
            ans += intersec(1, 3);
        printf("%lld\n", ans);
    }
    return 0;
}
```