E1 - Solution

A 带鱼除法

难度	考点
1	换行符，基本运算

题目分析

x 个带鱼，每天吃 y 个，按照整除的知识可知能吃 `x/y` 个，按照余数的知识可以吃 `x%y` 个，同时注意输出换行符 `\n` 。

本题不要用循环去模拟吃鱼的过程，多从数学方向去思考，而不是随便写循环去模拟。

示例代码

```c
#include <stdio.h>
int main()
{
    int x, y;
    scanf("%d%d", &x, &y);
    printf("%d\n", x / y);
    printf("%d", x % y);
    return 0;
}
```

B 简单的能量块

难度	考点
1	判断

题目分析

这是一道简单的能量块问题

可能需要注意的要点有：

1. 输入输出，还有格式问题
2. 变量初始化为 0 （如答案中记录 $\sum_+$ 的 `s1` 和记录 $\sum_-$ 的 `s2` ）
3. 判断条件的书写
4. 变量名只能由英文字母、数字、下划线组成，且第一个字符不能为数字，所以一定不要用希腊字母去命名变量哦~

示例代码

```
#include <stdio.h>
int main()
{
    int a;
    int s1 = 0,s2 = 0; //s1记录正数，s2记录负数
    for(int i = 1; i <= 4; ++i) //循环读入四个数
    {
        scanf("%d", &a);
        if(a > 0) s1 += a; //判断
        else s2 += a;
    }
    printf("%d\n%d", s1, s2); //输出
    return 0;
}
```

C 摩卡与小学期

难度	考点
1	分支结构

题目分析

根据题目要求，我们先要对预先给定个数的 `贡献值` 求和，然后根据题目描述和样例，进行一个判断：如果 `贡献值` 之和小于等于 0， 则 Moca 团队一个需求也没有完成；如果 `贡献值` 之和大于等于 `老师发布的需求数`，那么 Moca 团队完成了老师发布的全部需求；否则 Moca 团队完成了与贡献值之和相等的需求。

此题需要注意的就是一些特殊的数据点，这里故意设了几个数据点，包括 Moca 团队的贡献值之和**恰好等于** 0 ，Moca 团队的贡献值之和**恰好等于**老师发布的需求数，这两种情况下按照题意是要输出特殊语句的。有的同学忽略了临界情况，即在分支结构的条件判断中没有写等号，导致没能通过特殊数据点。

其次还有一点需要额外说明的是，一些同学用变量 `sum` 保存贡献值之和，在 $main$ 里定义了 `sum` 但是没有初始化，这种情况下会导致 `sum` 的初值不一定是 0，原因在此不过多赘述。总之，大家一定要注意类似于这样的问题，即应该写 `int sum = 0;` 。

示例代码

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int n, a, need;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        scanf("%d", &a);
        sum = sum + a;
```

```
    }
    scanf("%d", &need);

    if( sum <= 0 ) {
        printf("%d\n", 0);       // 根据题意，贡献值为负或者为 0 时是一个需求也没完成的情况
        printf("Moca finish 0 requirement!");
    }
    else if( sum >= need ) {
        printf("%d\n", need);    // 根据题意，贡献值大于等于老师发布的需求则完成了所有需求
        printf("Moca finish all requirements!");
    }
    else {
        printf("%d\n", sum);
    }

    return 0;
}
```

D czx 的出游计划

难度	考点
2	条件语句

题目分析

根据题目意思，先读入相应的数据，随后用 `if-else` 语句判断一下 op 是哪一种情况，开两个变量记录小牛的兴奋值和愉悦值（记得初始化），根据题意依次计算即可。

示例代码

```
#include <stdio.h>

int main() {
    int a1, a2, a3, b1, b2, b3, op;
    int x = 0, res = 0; //兴奋值和愉悦值
    scanf("%d%d%d%d%d%d%d", &a1, &b1, &a2, &b2, &a3, &b3, &op);
    if (op == 1) {
        x += a1, res += x * b1;
        x += a2, res += x * b2;
        x += a3, res += x * b3;
    } else if (op == 2) {
        x += a2, res += x * b2;
        x += a3, res += x * b3;
        x += a1, res += x * b1;
    } else if (op == 3) {
        x += a3, res += x * b3;
        x += a2, res += x * b2;
        x += a1, res += x * b1;
    }
    printf("%lld\n", res);
    return 0;
```

```
}
```

E 阶乘对对碰

难度	考点
2	循环 判断语句

题目分析

题目考察的是关于简单的循环语句和判断语句的使用。

首先，根据阶乘与双阶乘的公式，利用 for 循环求得最后 a 的阶乘和 b 的双阶乘。

然后，比较 a 的阶乘和 b 的双阶乘的大小并对其大小关系进行输出。

特别地，需要注意 $0!=0!!=1$ 。

示例代码

```c
#include <stdio.h>

int main()
{
    // 输入两个小于10的数
    int a, b;
    scanf("%d %d", &a, &b);

    // 计算a的阶乘和b的双阶乘
    int a_factorial = 1;

    for (int i = 1; i <= a; i++)
    {
        a_factorial *= i;
    }

    int b_double_factorial = 1;

    for (int i = b; i > 0; i -= 2)
    {
        b_double_factorial *= i;
    }

    printf("%d %d\n", a_factorial, b_double_factorial);

    // 比较两者的大小，并输出
    if (a_factorial > b_double_factorial)
    {
        printf("a!>b!!");
    }
    else if(a_factorial < b_double_factorial)
    {
        printf("a!<b!!");
```

```
    }
    else printf("a!=b!!");
    return 0;
}
```

F 入学考试

难度	考点
3	判断

问题分析

题意较好理解。我们需要读入所有学生的成绩，并把及格学生的成绩挑选出来单独计算。对每个学生的成绩进行判断是否及格，如果及格，则计入及格学生的总分中；同时，需要记录所有学生的总分。在判断的过程中记录及格人数和总人数，最后计算得到答案。

注意所有计数变量都需初始化。

参考代码

```
#include <stdio.h>

int main()
{
    int a, score, sum = 0, sumj = 0, cnt = 0, cntj = 0;
    scanf("%d%d", &a, &score);
    while (score != -1)
    {
        if (score >= a)
        {
            sumj += score;
            cntj++;
        }
        sum += score;
        cnt++;
        scanf("%d", &score);
    }
    printf("%d %d ", cntj, sum / cnt);
    if (cntj)
        printf("%d", sumj / cntj);
    else
        printf("-1");
    return 0;
}
```

G czx 与优购超市

难度	考点
3	条件语句

题目分析

每次操作，可以使用四个 `if-else` 语句确定 czx 的举动，然后先判断该次举动能否成功执行（钱够不够？商店里有没有剩余？czx 是否还拥有？）并输出相应的语句，如果成功了，还需要进行相应的更改（扣除一定的钱和商店里的剩余数量，扣除 czx 拥有的水果数量）。

在判断操作能否执行的时候，注意判断符号应当用 `<` 还是 `<=`。

在 std 里，两个商品的价格被设为了常量，这可以避免在打代码时手误打错，并使代码的可读性提高。同时也要注意自己使用的变量名尽量有意义，在使用变量较多时，一昧的定义 `a,b,c,d,e,...` 等意义不明显的变量，容易导致代码难以调试，出现“自己写的代码自己都看不懂”的情况。

示例代码

```
#include <stdio.h>

int main()
{
    const int priceA = 3, priceB = 2;
    int n, x, a, b, ownA = 0, ownB = 0;
    scanf("%d%d%d%d", &n, &x, &a, &b);
    for (int i = 1; i <= n; i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        if (u == 0)        // try to buy
        {
            if (v == 0)    // apple
            {
                if (x >= priceA && a > 0)
                {
                    a--;
                    x -= priceA;
                    ownA++;
                    puts("Ok!");
                }
                else
                    puts("Failed!");
            }
            else           // banana
            {
                if (x >= priceB && b > 0)
                {
                    b--;
                    x -= priceB;
                    ownB++;
                    puts("Ok!");
                }
```

```
                else
                    puts("Failed!");
            }
        }
        else              // try to eat
        {
            if (v == 0)   // apple
            {
                if (ownA > 0)
                {
                    ownA--;
                    puts("Ok!");
                }
                else
                    puts("Failed!");
            }
            else          // banana
            {
                if (ownB > 0)
                {
                    ownB--;
                    puts("Ok!");
                }
                else
                    puts("Failed!");
            }
        }
    }
    printf("%d %d %d", a, b, x);
    return 0;
}
```

H Ne：小杨掷骰子吗

难度	考点
4	循环、枚举

题目分析

题目要求判断是否存在一种情况，使得三个骰子的红色面之和为 A，且黑色面之和恰好为 B，我们考虑直接三重循环枚举所有情况，第一重为枚举第一个骰子的面朝向，第二和第三同理。

另外，我们假设两个数 `sum_A` 记录红色朝上的和， `sum_B` 记录黑色朝上的和。如果当前朝上的面是1或者4，那么我们就给 `sum_A` 加上这个数，如果不是，则给 `sum_B` 加上这个数。

那么，如果 `sum_A == A && sum_B == B` 则说明可以组成题目要求的结果，输出 `Yes` ，如果循环全部结束了还是没有判定成功的情况，则输出 `No`

示例代码

```
#include <stdio.h>

int main()
{
    int test;
    int a, b;
    scanf("%d", &test);
    while(test--)
    {
        scanf("%d %d", &a, &b);
        int jud = 0;
        for(int i = 1; i <= 6; i++)
            for(int j = 1; j <= 6; j++)
                for(int k = 1; k <= 6; k++)
                {
                    int A = 0, B = 0;
                    if(i == 1 || i == 4) A += i;
                    else B += i;
                    if(j == 1 || j == 4) A += j;
                    else B += j;
                    if(k == 1 || k == 4) A += k;
                    else B += k;
                    if(A == a && B == b)
                        jud = 1;
                }
        if(jud) printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

I 将军点兵

难度	考点
2(枚举法) or 4(中国剩余定理)	循环，判断，取模运算

题目分析

重述题目：本题给定三个除数 x_i 和对应的余数 y_i，求满足条件的被除数 $result$ 。

事实上，最笨的办法便是**从1开始**使用，使用循环找到满足条件的被除数，这个方法是最慢的（循环次数最多），但很显然本题作者允许你这样写了。使用这一思路的同学可以参看示例代码1。

那么有没有什么简化的办法呢？显然我们可以只去找那些被 x_1 除余 y_1 的数，再判断是否满足 x_2 与 x_3 的条件。这样我们从循环变量的角度减少循环次数。使用这一思路的同学可以参看示例代码2。

还有没有其他方法？我们可以使用如下公式来构造我们的被除数：

$$result = n_1 * (x_2 * x_3) + n_2 * (x_1 * x_3) + n_3 * (x_1 * x_2)$$

可以认为我们的被除数由被加号 + 三部分构成，三部分中，只有第一部分 $n_1 * (x_2 * x_3)$ 不能被 x_1 整除，于是我们要让这一部分满足被 x_1 除余 y_1 即可，类似地构造其余部分，需要注意我们应当对结果取模（$x_1 * x_2 * x_3$）。这样我们进一步简化循环，使用这一思路的同学你很棒，可以参看示例代码3。

Q&A

Q：为什么能简化循环，这样做有什么意义？

A：从示例代码1到3，我们能够发现循环过程中的循环变量更新的步长变大了，这也意味着我们预先排除的数变多了，从而可以简化循环，简化循环有助于减少运算次数，节约时间成本，另一个原因是后续的比赛中大家会经常见到"TLE"。

示例代码1

```
#include<stdio.h>
int main()
{
    //读取
    int m1, m2, m3; //除数
    int res1, res2, res3; //余数
    scanf("%d%d%d%d%d%d", &m1, &res1, &m2, &res2, &m3, &res3);
    //处理
    int mul = 1; //结果
    int flag = 0; //标志
    for(; flag != 1; mul++)
    {
        if(mul % m1 == res1)
        {
            if(mul % m2 == res2)
            {
                if(mul % m3 == res3)
                {
                    flag = 1;
                }
            }
        }
    }
    printf("%d", mul - 1); //由于使用for循环，跳出前会多自增一次，需要减去
    return 0;
}
```

示例代码2

```
#include<stdio.h>
int main()
{
    //读取
    int m1, m2, m3;
    int res1, res2, res3;
    scanf("%d%d%d%d%d%d", &m1, &res1, &m2, &res2, &m3, &res3);
    //处理
    int mul = res1;
    for(int flag = 0; flag != 1; mul += m1)
    {
```

```
        if(mul % m2 == res2)
        {
            if(mul % m3 == res3)
            {
                flag = 1;
            }
        }
    }
    printf("%d", mul - m1);
    return 0;
}
```

示例代码3

```
#include<stdio.h>
int main()
{
    //读取
    int m1, m2, m3;
    int res1, res2, res3;
    scanf("%d%d%d%d%d%d", &m1, &res1, &m2, &res2, &m3, &res3);
    //处理
    int mul = m1 * m2 * m3;
    int result = 0;
    int t1 = mul / m1, t2 = mul / m2, t3 = mul / m3;
    int tmp = t1 % m1; //当前循环变量
    int cnt = 1;
    while(tmp != 1) //先找到余1的
    {
        cnt++;
        tmp = (t1 % m1) * (cnt % m1) % m1;
    }
    result += (res1 * cnt % mul) * t1 % mul; //再利用乘法找到余res1的
    //使用取余运算防止乘法溢出
    //当然使用long long int 也可以。
    //以下类似
    tmp = t2 % m2;
    cnt = 1;
    while(tmp != 1)
    {
        cnt++;
        tmp = (t2 % m2) * (cnt % m2) % m2;
    }
    result = result % mul + (res2 * cnt % mul) * t2 % mul;
    tmp = t3 % m3;
    cnt = 1;
    while(tmp != 1)
    {
        cnt++;
        tmp = (t3 % m3) * (cnt % m3) % m3;
    }
    result = result % mul + (res3 * cnt % mul) * t3 % mul;
    //需要注意，此处得到的结果并不一定是最小的，需要取模得到最小的。
    printf("%d", result % mul);
    return 0;
```

```
}
```

J 基于快速数论变换的加密算法

难度	考点
3(枚举) or 5(解密算法)	循环，模运算

一方面，我们希望同学们观察加密算法的加密过程，熟悉模运算中简单的加法逆元与简单的乘法逆元，我们有：

$$
\begin{aligned}
x-1y &\equiv x+4y \pmod 5\\
x-2y &\equiv x+3y \pmod 5\\
x-3y &\equiv x+2y \pmod 5\\
x-4y &\equiv x+1y \pmod 5\\
\frac{1}{2}x &\equiv 3x \pmod 5\\
\frac{1}{3}x &\equiv 2x \pmod 5\\
\frac{1}{4}x &\equiv 4x \pmod 5
\end{aligned}
$$

注意，在 C 语言中，如果 $x<0$ 且 $y>0$，那么 `x % y` 的值一定不是正数。

在加密算法中，e,g 由 a,c 决定，f,h 由 b,d 决定，i,j 由 e,f，k,l 由 g,h 决定，所以我们可以根据二元一次方程组的求解得到以下代码：

```c
#include <stdio.h>
int main(void) {
    int i, j, k, l;
    scanf("%d%d%d%d", &i, &j, &k, &l);
    int g = (k + 1 * l) * 3 % 5;
    int h = (k + 4 * l) * 4 % 5;
    int e = (i + 1 * j) * 3 % 5;
    int f = (i + 4 * j) * 3 % 5;
    int a = (e + 1 * g) * 3 % 5;
    int b = (f + 1 * h) * 3 % 5;
    int c = (e + 4 * g) * 3 % 5;
    int d = (f + 4 * h) * 3 % 5;
    printf("%d %d %d %d", a, b, c, d);
    return 0;
}
```

另一方面，我们希望同学们不观察加密算法的加密过程，熟悉 `for` 循环与 `if` 选择。

由于加密前 a,b,c,d 都是小于 5 的自然数，我们可以使用 `for` 循环枚举全部可能的 `a`, `b`, `c`, `d`，直接使用题目给出的加密算法进行加密，然后使用 `if` 选择符合条件的情况。这样我们就得到了如下代码：

```c
#include <stdio.h>
int main() {
    int m, n, o, p;
```

```
    scanf("%d%d%d%d", &m, &n, &o, &p);
    int a, b, c, d;
    for (a = 0; a < 5; a++) {
        for (b = 0; b < 5; b++) {
            for (c = 0; c < 5; c++) {
                for (d = 0; d < 5; d++) {
                    int e = (a + 1 * c) % 5;
                    int f = (b + 1 * d) % 5;
                    int g = (a + 4 * c) % 5;
                    int h = (b + 4 * d) % 5;
                    int i = (e + 1 * f) % 5;
                    int j = (e + 4 * f) % 5;
                    int k = (g + 2 * h) % 5;
                    int l = (g + 3 * h) % 5;
                    if (i == m && j == n && k == o && l == p) {
                        printf("%d %d %d %d", a, b, c, d);
                    }
                }
            }
        }
    }
    return 0;
}
```

我们建议学有余力的同学阅读题目中的思考部分。

- End -