C1 - Solution

A 教师节快乐

难度	考点
1	字符串输出

题目分析

输出如题所示的字符串即可，可以直接复制粘贴。

示例代码

```c
#include <stdio.h>

int main() {
    printf("Dear teacher, I love yOu.");
    return 0;
}
```

B 献给老师的小红花

难度	考点
2	判断结构 基本运算

题目分析

如题意所示，输出两行，分别是 k / n （正如我们在课上学到过的，两个正整数的除法结果还是一个整数，相当于对我们日常生活中知道的除法向下取整）和 $k \ mod \ n$ 。要注意的是，输出多行字符串，需要加上换行符，否则输出会挤在同一行。

错误示范

```c
printf("%d", k / n);
printf("%d", k % n);
```

此时输入 2 和 5 ，得到输出

```
21
```

正确的做法：加上换行符

```c
printf("%d\n", k / n);
printf("%d\n", k % n);
```

当 $n = 0$ 时，输出特殊的结果（这个时候再做除法或取余运算相当于除数是 0，所以程序就会出现一个异常），整体程序可以参考 PPT 中的例 1 - 6，可以得到下面的示例代码。

示例代码

```c
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d%d", &n, &k);
    if(n == 0) {
        printf("Wait a minute\n");
    }
    else {
        printf("%d\n", k / n);
        printf("%d\n", k % n);
    }
    return 0;
}
```

扩展补充

这里补充一下 C 语言的整数除法和取余运算，我们知道比如在同余的角度下，$4 \bmod 3 = 1$ 或者 $4 \bmod 3 = -2$ 都可以说是对的，那么 C 语言是如何规定取余操作的呢？可以尝试以下的代码片段：

```c
#include <stdio.h>

int main()
{
    printf("4 %% 3 = %d\n",4 % 3);
    printf("-2 %% 3 = %d\n",(-2) % 3);
    printf("-5 %% 3 = %d\n",(-5) % 3);
    printf("(-2 + 3) %% 3 = %d\n",(-2+3) % 3);
    printf("(-5 + 3) %% 3 = %d\n",(-5+3) % 3);
    printf("(-2 %% 3 + 3) %% 3 = %d\n",(-2%3 + 3) % 3);
    printf("(-5 %% 3 + 3) %% 3 = %d\n",(-5%3 + 3) % 3);

    printf("\n");

    printf("4 %% -3 = %d\n",4 % -3);
    printf("-2 %% -3 = %d\n",(-2) % -3);
    printf("-5 %% -3 = %d\n",(-5) % -3);
    printf("(-2 + 3) %% -3 = %d\n",(-2+3) % -3);
    printf("(-5 + 3) %% -3 = %d\n",(-5+3) % -3);
    printf("(-2 %% -3 + 3) %% -3 = %d\n",(-2%3 + 3) % -3);
    printf("(-5 %% -3 + 3) %% -3 = %d\n",(-5%3 + 3) % -3);
    return 0;
}
```

在 C 语言中，余数的计算方法可以表示为 $a \bmod b = a - a/b * b$ 。而 C 语言中除法是趋零截尾的，也就是整数除法的结果相当于除法的结果向 0 的方向，将小数部分阶段，比如计算整数除法 $7/(-3)$，$\frac{7}{-3} = -2.33$，所以整数除法结果就是 -2，即直接截断了小数点后的部分。

所以余数和被除数的符号一致，也就是说余数并不总是和通常的取余运算结果相一致（通常数学上的取余操作 $a \bmod b$ 结果在 0 到 $b-1$），所以在执行完 `ans = a % b` 后，为了使结果 ans 一定在 0 到 $b-1$ 之间，我们用 `ans = (ans + b) % b`。这样，如果 ans 以前的值就是一个正数，那么 ans 的值不会受到影响；如果 ans 的值之前是一个负数，这么做就相当于把它转化成了同余意义下等价的正数，符合我们通常数学中的取余操作。

C 拼接URL

难度	考点
1	字符串输出

题目分析

读入数字，按照要求依次输出即可。

注意：C 语言不像 Python 一样，可以直接对字符串和整数使用加法操作

示例代码

```c
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>

int main(){
    int a;
    scanf("%d",&a);
    printf("https://sns-img-bd.brdcdn.com/%d",a);
    return 0;
}
```

D 摩卡与分数统计

难度	考点
2	基本运算，分支结构

题目分析

由题知，先把题目中输入的三个正整数相加，如果超过 100 输出 100；否则输出相加后的和。则先参考 PPT 上的例 1-2，将三数相加，再参考例 1-5，写一个简单的分支结构

示例代码

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

```c
int main() {
    int sum = 0;
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);

    sum = a + b + c;

    if(sum >= 100) {
        printf("100");
    }
    else {
        printf("%d", sum);
    }

    return 0;
}
```

E 小懒獭与叉积

难度	考点
2	四则运算

题目分析

根据题目中给出的公式，即三维向量的叉积：

$$\overrightarrow{a} \times \overrightarrow{b} = (a_2b_3 - a_3b_2,\ a_3b_1 - a_1b_3,\ a_1b_2 - a_2b_1)$$

依次读入六个整数 $a1\ a2\ a3\ b1\ b2\ b3$ 后，依次输出 $a_2b_3 - a_3b_2$ ，$a_3b_1 - a_1b_3$ 和 $a_1b_2 - a_2b_1$ 即可，中间用空格隔开。

示例代码

```c
int main() {
    int sum = 0;
    int a1, a2, a3;
    int b1, b2, b3;
    scanf("%d%d%d", &a1, &a2, &a3);
    scanf("%d%d%d", &b1, &b2, &b3);

    printf("%d ", a2 * b3 - a3 * b2);
    printf("%d ", a3 * b1 - a1 * b3);
    printf("%d", a1 * b2 - a2 * b1);

    return 0;
}
```

F 小水獭与月饼

难度	考点
2	循环结构，数学运算

题目分析

参考 PPT 例 1-8 的代码，几乎是没变的

示例代码

```c
#include <stdio.h>

int main() {
    int sum = 0, n = 0; // 表示月饼总数和水獭的数目
    int num;    // 表示读入的当前水獭的月饼数

    scanf("%d", &num);       // 输入第一个月饼数

    while(num != -1) {
        sum = sum + num;     // 月饼累加
        n = n + 1;           // 水獭数累加
        scanf("%d", &num);   // 输入下一个月饼数
    }

    printf("%d", sum / n);     // 两个正整数之间除法向下取整

    return 0;
}
```

扩展延伸

想一想，如果要问的是重新分配后，拥有月饼数最多的水獭应该如何计算？

G 摩卡与水獭餐厅

难度	考点
2	分支结构 循环结构

题目分析

根据题意，首先应该判断来的水獭数是不是大于餐厅座位数，即：

```c
if(n > k) {
    do something ...
} else {
    do something ...
}
```

如果水獭数多于座位数，直接输出题目中要求的字符串；否则，参考代码例 1-7，输出 n 个 `*` 后（每个后面有空格），剩下的座位数是 $k-n$ 个，输出 $k-n$ 个 `_` （后面带空格）。

示例代码

```
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d%d", &n, &k);

    if(n > k)
        printf("Too many little otters!");
    else {
        for(int i = 1; i <= n ;i++) {
            printf("* ");
        }
        for(int i = n + 1; i <= k ;i++) {
           printf("_ ");
        }
    }

    return 0;
}
```

H 小P的乌龟爬水井

难度	考点
2	贪心、分类

题目分析

此题比较简单，若位于较前位置，在时间充沛的情况下，大约有 80% 的同学可以通过，预估平均分 $80/100$ 。

由题，我们可以发现，当 $a \geq h$ 时，乌龟总能在第 1 天到井口，故可将此类情况单独分出。

之后，当 $a < h$ 时，可再分为两种情况。

1. 当 $a \leq b$ 时，由于乌龟上升距离不大于下降距离，故其每天开始时总位于井底。又由于其不能一次爬到井口，即 $a < h$ ，故其始终不能到达井口。
2. 当 $a > b$ 时，由于乌龟上升距离大于下降距离，故其每天的开始高度总比上一天高。故而，其总能到达井口。并且，由于夜晚的高度会下降，故乌龟一定在最后一天的白天就到达了终点。设第 ans 到达了井口，则一定有 $(ans-1)*(a-b)+a \geq h$ 其中 $(ans-1)*(a-b)$ 代表前 $ans-1$ 天行进的路程，而 a 代表最后一天行进的路程，二者相加应大于 h。变形可得 $ans \geq (h-a)/(a-b)+1$，即 $ans = \lceil (h-a)/(a-b) \rceil + 1$ ，直接计算即可。

具体代码见下：

示例代码

std：

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
signed main()
{
    int a,b,h,ans;//此题中int就足够了
    scanf("%d%d%d",&h,&a,&b);
    if(a>=h)//a>=h时，第一天白天就可到顶，与b无关
    {
        printf("1\n");
    }
    else if(a<=b)//a<h且a<=b时，总无法到顶
    {
        printf("Impossible\n");
    }
    else//b<a<h时，可通过表达式计算出天数
    {
        ans=((h-a)+(a-b-1))/(a-b)+1;//a/b向上取整等价于(a+b-1)/b的向下取整
        printf("%d\n",ans);
    }
    return 0;
}
```

I 军乐团破冰

难度	考点
3	最大公因数 gcd、最小公倍数 lcm

题目分析

分析题意不难发现，答案就是 8 个连队参加人数的最小公倍数 $*8$。
$gcd(a,b)$ 为 a 和 b 的最大公因数，$lcm(a,b)$ 为 a 和 b 的最小公倍数，易证：
$gcd(a,b,c)=gcd(\ gcd(a,b)\ ,c)$

$lcm(a,b,c)=lcm(\ lcm(a,b)\ ,c)$
由此可以得出：
$lcm(\text{前 } i \text{ 个数})=lcm(\ lcm(\text{前 } i-1 \text{ 个数})\ ,\ \text{第 } i \text{ 个数})$
因此依次计算前 $2,3,4,\ldots,8$ 个数的 lcm 即可得到答案，但是 lcm 可能会很大，如果从 1 向上枚举很有可能超时，这时就要用到：
$lcm(a,b)=\frac{a*b}{gcd(a,b)}$
因为 $gcd(a,b)\le \min(a,b)$，所以每轮 gcd 的值可以从 200 向 1 枚举，这样就不会超时啦！

示例代码

```
#include<stdio.h>
int main()
{
    int lcm,a;
    scanf("%d",&lcm); //第1个数
    for(int i=1;i<=7;i++)
    {
        scanf("%d",&a); //后7个数
        for(int gcd=200;gcd>=1;gcd--)  //从高向低枚举gcd
        {
            if(lcm%gcd==0&&a%gcd==0) //找到gcd了
            {
                lcm/=gcd; //更新lcm,注意此处写法!!!
                lcm*=a;
                break; //循环提前结束
            }
        }
    }
    printf("%d",lcm*8); // 别忘了最后*8
    return 0;
}
```

注意

示例代码中更新 lcm 时语句有多种写法，其中一种错误写法是：

```
int lcm;
lcm=lcm*a/gcd;
```

错误之处在于尽管题目保证了 lcm 在 `int` 范围内，但 $lcm*a$ 的值很可能超过 `int` 范围，先计算 `*a` 再计算 `/gcd` 很可能导致局部溢出超出 `int` 范围，大家可以上网搜索局部溢出的后果，这里不详细介绍了。为了改正这个错误，可以将 lcm 改成 `long long` 类型，或者选取先计算 `/gcd` 再计算 `*a` 的写法。

补充内容

对于本题的数据范围 $1 \leq a_i \leq 200$ 且只有 8 个数，暴力遍历 gcd 时间似乎是足够的，但对于更大规模的求 gcd，有没有更高效的方法呢？

这里贴出一个古老且神奇的 **辗转相除法**，感兴趣的同学可以自行研究它的原理和正确性，对于求 $gcd(a,b)$，其时间复杂度从 $O(n)$ 降到了 $O(\log n)$，$n=\min(a,b)$。（$\log n$ 表示数学中的 $\log_2 n$）

```
int a,b,gcd,temp;  //求正整数a和b的最大公因数
scanf("%d%d",&a,&b);
while(b!=0)
{
    temp=a%b;
    a=b;
    b=temp;
}
gcd=a;
printf("%d",gcd); //最终求出gcd的值
```

J 摩卡与探险家水獭

难度	考点
4	数学

题目分析

看到这道题感觉很难观察出什么结果，但是毕竟把递推关系给了我们，我们不妨先列出前几项试试看：

```
a1 = 1
a2 = 3
a3 = 6
a4 = 10
...
```

写出前几项后，我们惊奇地发现，这几个数字似乎都满足 $a_m = \frac{m(m+1)}{2}$ 。我们能否基于这个猜想，去反过来证明这个公式是对的呢？

根据第二数学归纳法：

- 当 $m = 1$ 时，有 $a_1 = 1 \times (1+1)/2 = 1$ 成立
- 对 $n > 1$ ，假设所有的正整数 $k < n$，都有 $a_k = k(k+1)/2$ 成立，注意到：

$$\frac{i(i+1)}{2} + \frac{j(j+1)}{2} + ij = \frac{(i+j)(i+j+1)}{2}$$

故

$$a_n = \max\{a_k + a_{n-k} + k(n-k) \mid 1 \leqslant k < n\} = \max\left\{\frac{(k+n-k)(k+n-k+1)}{2} \mid 1 \leqslant k < n\right\}$$

即 $a_n = \frac{n(n+1)}{2}$

由第二归纳法原理，有 $a_m = \frac{m(m+1)}{2}$ 成立。

示例代码

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    printf("%d", n * (n + 1) / 2);

    return 0;
}
```

扩展延伸

当我们做不上一道题的时候，不妨自己手动（或者用程序暴力）模拟一下几个小数据的结果，观察一下有什么规律，说不定会有意外的发现！（虽然这种方法并不总是有效的）