A - 提示の痛

难度	考点
1	读题

题目分析

本题主要是为了强调提示的重要性，把下面提示里的公式照搬到程序里就行了。

注意：需要判断“库默-艾诺条件”的不是输入的数字，而是“metricolla 函数”的计算结果。再次强调读题要仔细。

示例程序

```c
#include <stdio.h>

int main()
{
    int a, r;

    scanf("%d", &a);
    r = (a ^ 114514) >> 2;
    printf("%d\n", r);
    if (((r + 521) & 1) > 0)
    {
        printf("I can see the hint");
    }
    else
    {
        printf("HINT!");
    }

    return 0;
}
```

B - 位运算比大小（简单版）

难度	考点
2	二进制

题目分析

本题的题意较为直白，同学们只需要根据题目要求，直接二重循环遍历所有可能的无序二元组并判断即可，注意运算符优先级。

示例程序

```c
#include <stdio.h>

int main()
{
    int a[5000 + 5];
    int n, i, j, ans = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }
    for (i = 0; i < n - 1; i++)
    {
        for (j = i + 1; j < n; j++) // 注意从 i+1 开始避免重复计算 (i, j) 和 (j, i)
        {
            if ((a[i] & a[j]) > (a[i] ^ a[j])) // 注意此处适当添加括号
                ans++;
        }
    }
    printf("%d", ans);

    return 0;
}
```

C - 301 Moved Permanently!

难度	考点
2	重定向

题目分析

本题主要考察重定向，可以参考第三次课件中的 `freopen` 的使用方法。

需要注意的是，题目要求最后一组数据末尾不输出换行符，如果输出了多余换行符可能导致 WA。

示例程序

```c
#include <stdio.h>

int main()
{
    freopen("301.txt", "w", stdout);

    int n, op, i;
    long long a1, b1;
    double a2, b2;

    scanf("%d", &n);
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &op);
        if (op == 1 || op == 2)
        {
            scanf("%lld%lld", &a1, &b1);
            printf("%lld", a1 + b1);
        }
        else
        {
            scanf("%lf%lf", &a2, &b2);
            printf("%.4f", a2 + b2);
        }
        if (i != n) printf("\n");
    }

    return 0;
}
```

D - DECIMAL or TERNARY?

难度	考点
3	数学计算

题目分析

将一个数字变成三进制，只需要对它逐步模三取余再除以三直到它变成 0，再倒序输出即可。

示例程序

```c
#include <stdio.h>

int main()
{
    int a;
    int ans[100], num, i;

    while (scanf("%d", &a) > 0)
    {
        num = 0;
        ans[num] = a % 3;
        a /= 3;
        num++;
        while (a > 0)
        {
            ans[num] = a % 3;
            a /= 3;
            num++;
        }
        if (num >= 5)
        {
            printf("LONG");
        }
        for (i = num - 1; i >= 0; --i)
        {
            printf("%d", ans[i]);
        }
        printf("\n");
    }

    return 0;
}
```

E - 刷朋友圈

难度	考点
3	模拟

题目分析

本题需耐心读题，配合样例准确理解题意。之后按照题目的规则用循环按顺序模拟一遍刷朋友圈的过程即可。

示例程序

```c
#include <stdio.h>

int main()
{
    int n, L, i, sum = 0, l, ans = 1;

    scanf("%d%d", &L, &n);
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &l);
        sum += l;
        if (sum > L)
        {
            ans += 2;
            sum = 0;
            if (i == n) ans--;
        }
        else if (sum == L)
        {
            sum = 0;
            ans++;
            if (i == n) ans--;
        }
    }
    printf("%d", ans);

    return 0;
}
```

F - 0 和 1

难度	考点
3	二进制

题目分析

本题要求较为直白，只要用二重循环分别依次求出范围内每个数字的 0 和 1 的数量即可。

注意在统计时，如果 $L=0$，要额外处理一下，否则会少算一个 0。

示例程序

```c
#include <stdio.h>

int main()
{
    int l, r, temp;
    int zero = 0, one = 0;

    scanf("%d%d", &l, &r);
    if(l == 0) zero++;

    for (; l <= r; l++)
    {
        for (temp = l; temp; temp >>= 1)
        {
            if (temp & 1)
                one++;
            else
                zero++;
        }
    }
    printf("%d %d\n", zero, one);

    return 0;
}
```

G - 我要当矿主

难度	考点
3	补码 位运算

题目分析

这道题实际上是一道求补码题。对于给定的变量，我们遍历它的 32 位补码表示中的每一位，统计 0 的个数并与 $targetbits$ 比较即可。C 语言中，整型变量的就是以补码的形式存储在内存中的，我们只需要利用位运算使每一位 `& 1`，即可遍历变量的补码。具体代码如下：

示例程序

```c
#include <stdio.h>

int main()
{
    int t;
    int i;
    int hash;
    int sum = 0;

    scanf("%d", &t);
    scanf("%d", &hash);
    for (i = 0; i < 32; i++)
    {
        if (((hash >> i) & 1) == 0) sum++;
    }
    if (sum == t)
        printf("coins++");
    else
        printf("continue to work hard");

    return 0;
}
```

H - 位运算与集合

难度	考点
4	位运算、二进制、集合

题目分析

首先根据题目中给的信息，将 A, B 集合保存成一个整数，交集、并集、对称差则对应二进制运算中的与、或、异或。

字母 `alpha` 的编号可以利用 ASCII 计算，`alpha-'a'` 可以将 `a~z` 编号为 `0~25`。

示例程序

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

int main()
{
    int a, b, i, j, size;
    char sa[105], sb[105];

    while (~scanf("%s%s", sa, sb))
    {
        a = 0;
        b = 0;
        // 读入a
        for (i = 0; i < strlen(sa); i++)
        {
            a |= 1 << (sa[i] - 'a');
        }
        // 读入b
        for (i = 0; i < strlen(sb); i++)
        {
            b |= 1 << (sb[i] - 'a');
        }

        // 输出a
        for (i = 0; i < 26; i++)
        {
            if (a & (1 << i))
            {
                putchar(i + 'a');
            }
        }
        putchar('\n');

        // 输出b
        for (i = 0; i < 26; i++)
        {
            if (b & (1 << i))
            {
                putchar(i + 'a');
```

```
            }
        }
        putchar('\n');

        // 输出a、b交集
        for (i = 0; i < 26; i++)
        {
            if ((a & b) & (1 << i))
            {
                putchar(i + 'a');
            }
        }
        putchar('\n');

        // 输出a、b并集
        for (i = 0; i < 26; i++)
        {
            if ((a | b) & (1 << i))
            {
                putchar(i + 'a');
            }
        }
        putchar('\n');

        // 输出a、b对称差
        for (i = 0; i < 26; i++)
        {
            if ((a ^ b) & (1 << i))
            {
                putchar(i + 'a');
            }
        }
        putchar('\n');
    }

    return 0;
}
```

I - 寻找单身狗

难度	考点
6	位运算

知识点分析

本题主要考察了位运算中的异或运算，其中，异或运算的真值表如下：

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

我们可以发现，对于一位二进制数字，两个相同的数字异或结果为0。推广到int型变量，我们能够得到：$a \oplus a = 0$, $a \oplus 0 = a$。同时，异或运算是具有交换律和结合律的。掌握了这些后，这道题就可以做啦。

题目分析

1. 对于没有单身狗的情况，只需要遍历一遍，维护最大值最小值，输出即可。
2. 对于一只单身狗的情况，同样只需要遍历一遍，把所有数字做异或操作，最后得到的答案就是结果。
3. 对于两只单身狗的情况，需要遍历两遍，第一遍遍历把所有数字做异或操作，得到的结果是 $x = a \oplus b$，因为 $a \neq b$，所以结果的二进制中至少有一位为 1，我们不妨找到最低位的 1，不妨设 a 的这一位为 1，b 的这一位为 0。我们再遍历一次数组，把这一位为 1 的所有数字做异或运算，得到的结果就是 a，此时 $b = a \oplus x$ $(x = a \oplus b)$。

注意：对于三种情况，要用 `if` 条件语句分清。第三种情况两个数字的乘积可能会超过 `int` 范围，请用 `long long` 输出。

示例程序

```c
#include <stdio.h>
#define MaxInt 2147483647

int array[1000005];

int main()
{

    int ans = 0, max = 0, min = MaxInt, a, b; // 全局变量初始值为 0
    int N, temp, flag, position = 0, i, ans1;

    scanf("%d", &N);
    for (i = 0; i < N; i++)
```

```
    {
        scanf("%d", &array[i]);
        temp = array[i];
        ans = ans ^ temp;
        if (temp > max) max = temp;
        if (temp < min) min = temp;
    }
    if (ans == 0 && (N & 1) == 0)
    {
        printf("%d %d", min, max);
    }
    else if ((N & 1) == 1)
    {
        printf("1 %d", ans);
    }
    else
    {
        ans1 = ans;             // 将ans保存下来，以便后面求b
        while ((ans & 1) == 0) // 求a^b二进制中最低位为1的，则a和b对应的位置一定一个是1，另一个是0
        {
            ans = ans >> 1;
            position++;
        }
        for (i = 0; i < N; i++) // 重新遍历数组，只要该位是1，就进行异或运算，最后得到的是a
        {
            if ((array[i] >> position) & 1 == 1)
            {
                a = a ^ array[i];
            }
        }
        b = a ^ ans1; // 因为ans1 = a ^ b, 所以 b = an1 ^ a
        printf("2 %lld", (long long)a * (long long)b);
    }

    return 0;
}
```

J - 基础物理实验（困难版）

难度	考点
7	数学思维，一维数组的运用

题目分析

我们可以将序列 $A=[p_1,p_2,\cdots,p_n]$ 按照下标的奇偶分为两个序列 $B=[p_1,p_3,\cdots]$ 和 $C=[p_2,p_4,\cdots]$，那么对于题目中的一次操作，等价于将上述任意一个序列中任意两个相邻的数字交换，因此题目有解的**第一个**必要条件是所有的奇数都出现在 B 中且所有的偶数都出现在 C 中。

我们定义一个序列的逆序对为

$$\{(i,j)\mid i<j \text{ and } p_i>p_j\}$$

因为每次操作要求 $p_{i-1}>p_i>p_{i+1}$，所以每次操作会令 A 中逆序对数量减少 3，B 和 C 中的恰好一个序列的逆序对数量减少 1。因此题目有解的**第二个**必要条件是 A 中逆序对数量是 B 中逆序对数量与 C 中逆序对数量是之和的三倍。

下面我们证明题目有解的充分条件是同时满足上述两个必要条件：因为我们满足**第一个**必要条件，所以可以考虑 `基础物理实验（简单版）` 中将两个 B 和 C 两个序列变为有序的过程。我们只需证明如果满足**第二个**必要条件的情况下，那么这个过程每一步都是合法操作（即满足 $p_{i-1}>p_i>p_{i+1}$），即可证明题目有解。

因为每次操作要求 $p_{i-1}>p_i>p_{i+1}$，所以每次操作如果 A 中逆序对数量减少 3 那么它一定是合法操作，同时每次操作 A 中逆序对数量至多减少 3 。而因为 A 中逆序对数量是 B 中逆序对数量与 C 中逆序对数量是之和的三倍，假设有一次操作不满足 A 中逆序对数量减少 3，那么最终当 B 和 C 都有序时，A 中逆序对数量必然大于 0，从而 A 无序，这与 B 和 C 都有序矛盾。因此每次操作会使 A 中逆序对数量减少 3，从而这个过程中每次操作都是合法操作。

由上述证明过程我们可以得到，如果题目有解，那么所需操作数为 B 中逆序对数量与 C 中逆序对数量之和。

我们可以使用一维数组和二重 `for` 循环来实现求解一个序列的逆序对数量，时间复杂度为 $O(n^2)$。

示例程序

```c
#include <stdio.h>

int main()
{
    int a[1086], t, n, tag, i, j;
    int x, y;

    scanf("%d", &t);
    while (t--)
    {
        scanf("%d", &n);
        for (i = 1; i <= n; i++)
            scanf("%d", &a[i]);
        tag = 0;
        for (i = 1; i <= n; i++)
            if ((i - a[i]) & 1) tag = 1;
```

```
        if (tag)
        {
            puts("-1");
            continue;
        }
        x = 0, y = 0;
        for (i = 1; i <= n; i++)
        {
            for (j = 1; j < i; j++)
            {
                x += a[j] > a[i];
            }
        }
        for (i = 1; i <= n; i += 2)
        {
            for (j = 1; j < i; j += 2)
            {
                y += a[j] > a[i];
            }
        }
        for (i = 2; i <= n; i += 2)
        {
            for (j = 2; j < i; j += 2)
            {
                y += a[j] > a[i];
            }
        }
        if (y * 3 != x)
        {
            puts("-1");
            continue;
        }
        printf("%d\n", y);
    }

    return 0;
}
```