E4 - Solution

A 信类友谊赛

难度	考点
1	模拟、条件判断、循环结构

题目分析

根据题意，我们需要维护一个传源书院和士谔书院的总胜局数。每轮比赛会给出两个得分 X_i 和 Y_i，分别表示传源书院和士谔书院在这一轮的得分：

- 如果 $X_i > Y_i$，传源书院胜局数加 1；
- 如果 $X_i < Y_i$，士谔书院胜局数加 1；
- 如果 $X_i == Y_i$，两个书院胜局数不变。

比赛结束后，通过对比两所书院的胜局数，输出对应的结果。

示例代码

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

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

    // 每一轮的比分情况来维护胜局
    for(int i = 1; i <= n ;i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        if(x > y)
            a++;
        else if(x < y)
            b++;
    }

    // 判断总的胜局情况
    if(a > b)
        printf("Chuanyuan wins");
    else if(a < b)
        printf("Shie wins");
    else
        printf("Draw");

    return 0;
}
```

B 摩卡与音游

难度	考点
2	模拟、循环、分支

题目分析

这题是一道简单的模拟题，数据范围发现表现分最大只有 6 亿，所以 int 就可以。

需要我们考虑的有两件事：如何获得最大连击数，如何知道最后是 `All Perfect!` 、`Full Combo!` 还是 `Moca Complete!` 。

对于第一个问题，我们可以用一个变量 `maxCombo` 来维护最大连击数，这个变量的初值为 0；用 `combo` 维护当前的连击数，每当获得一个 `p` 或者 `g` 时，`combo` 数增加 1；每当获得一个 `b` 或者 `m` 时，`combo` 数变为 0，但在 `combo` 数变为 0 之前，我们需要判断一下这次的连击数是不是大于最大连击数，如果是的话则需要更新 `maxCombo` 。这里需要注意的是如果只在断连时更新 `maxCombo` ，那么曲子结束之前最后一次的连击数就无法被用来更新 `maxCombo` 了，这里我们需要注意一下。

第二个问题的解决方法是多样的，可以用 `maxCombo` 来判断是不是 `Moca Complete!` （如果 `maxCombo` 与总音符数不相等那么就是 `Moca Complete!` ）；可以用一个标记来标记是否出想过 `g` ，以此来判断是 `All Perfect!` 还是 `Full Combo!` 。

示例代码

```
#include <stdio.h>

int main()
{
    int n;
    int score = 0;
    int maxCombo = 0;
    int combo = 0;
    int ap = 1; // 如果为 1 ，证明没有出现过 ap 以外的字符

    char c;
    int a,b;
    scanf("%d",&n);
    for(int i = 1;i <= n;i++)
    {
        getchar();  // 处理换行符
        scanf("%c",&c);
        switch (c)
        {
            case 'p':
                combo++;
                score += 300;
                break;

            case 'g':
                combo++;
                ap = 0;
                score += 208;
                break;
```

```
            case 'b':
                score += 105;
                if(combo > maxCombo) maxCombo = combo;
                combo = 0;
                ap = 0;
                break;

            case 'm':
                if(combo > maxCombo) maxCombo = combo;
                combo = 0;
                ap = 0;
                break;
        }
    }
    if(combo > maxCombo) {
        maxCombo = combo;
    }

    printf("%d\n%d\n",score,maxCombo);
    if(ap == 1) {
        printf("All Perfect!");
    }
    else if(maxCombo == n) {
        printf("Full Combo!");
    }
    else {
        printf("Moca Complete!");
    }
    return 0;
}
```

你也可以利用 `switch` 的一些特性将代码写的更简洁一些，但是在这里我们将不会给出这样的代码，以免带偏同学们。

C 不讲 5 的（小数据）

难度	考点
2	循环

题目分析

由于 n 的范围只有 10^6，因此我们直接枚举所有的有序数对 $(a, n-a)$ 即可。

还有一种更快的方法，我们可以将 0 到 n 范围内每个数是否含有数字 5 存储起来，这样就避免每次都判断一遍。并且注意到，对于 $k(k \geq 2)$ 位数而言，它没有数字 5，当且仅当它的前 $k-1$ 位没有数字 5 且它的个位不是 5。当我们计算这个数的时候，它的前 $k-1$ 位实际上已经计算过了，因此可以直接使用。

示例代码 1

```c
#include <stdio.h>

int main(void)
{
    int n, ans = 0;
    scanf("%d", &n);

    for (int i = 0; i <= n; i++)
    {
        int tem1 = i, tem2 = n - i;
        int valid1 = 1, valid2 = 1;

        // 检查 i 是否包含数字 5
        while (tem1)
        {
            if (tem1 % 10 == 5)
            {
                valid1 = 0;
                break;
            }
            tem1 /= 10;
        }

        // 检查 n - i 是否包含数字 5
        while (tem2)
        {
            if (tem2 % 10 == 5)
            {
                valid2 = 0;
                break;
            }
            tem2 /= 10;
        }

        // 如果 i 和 n - i 都不包含数字 5，增加计数
        if (valid1 && valid2)
            ans++;
    }

    printf("%d", ans);
    return 0;
}
```

示例代码 2

```c
#include<stdio.h>

int n,ans;
int check[1008611];
int main(){
    scanf("%d",&n);
```

```
    for ( int i=0 ; i<=9 ; i++ )
        check[i]=1;
    check[5]=0;
    for ( int i=10 ; i<=n ; i++ )
        check[i]=check[i/10]&&check[i%10];
    for ( int i=0 ; i<=n ; i++ )
        if ( check[i] && check[n-i] )
            ans++;
    printf("%d",ans);
    return 0;
}
```

D 摩卡与成绩统计 4

难度	考点
3	数组、分支结构

题目分析

本题实际上是含特判的多分支题目。我们的思路是可以先求出每只水獭的排名，再根据总水獭数判定某水獭获得的奖项（当然合并为一次循环操作也是可以的）。

我们注意到所给分数按照降序排列，因此本题可以通过**如果当前水獭的得分与上一位相同，则（进入特判）该水獭的排名和上一只水獭相同**来先处理排名。

在后续根据排名判断名次的区间最好使用整型变量判断，不会有误差。如 $r \leq n \cdot 10\%$ 等价于 $10 \times r \leq n$。后续的多分支情况注意使用 `else` 使各情况互斥即可。

示例代码

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

int rank[500005];
int score[500005];

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

    scanf("%d", &score[1]);

    rank[1] = 1;
    for(int i = 2; i <= n ;i++) {
        scanf("%d", &score[i]);
        if(score[i] == score[i - 1])
            rank[i] = rank[i - 1];
        else
            rank[i] = i;
    }
```

```
    for(int i = 1; i <= n ;i++) {
        if(rank[i] == 1)
            printf("Gold Award\n");
        else if(rank[i] == 2)
            printf("Silver Award\n");
        else if(rank[i] == 3)
            printf("Bronze Award\n");
        else if(10 * rank[i] <= n)
            printf("First Class Award\n");
        else if(10 * rank[i] <= n * 3)
            printf("Second Class Award\n");
        else if(10 * rank[i] <= n * 6)
            printf("Third Class Award\n");
        else if(10 * rank[i] <= n * 9)
            printf("Excellence Award\n");
        else
            printf("No Award\n");
    }

    return 0;
}
```

E 摩卡与 Cache

难度	考点
4	数组、模拟

题目分析

对于每次输入，我们无非有以下三种情况：

- **输入的数字在 Cache 中：** 遍历 Cache 数组，能找到这个数字，输出对应的位置
- **输入的数字不在 Cache 中，且 Cache 未满：** 遍历 Cache 数组，找不到这个数字，且已经占用的 Cache 大小没到 C，可以把这个元素存到已经存储的一段连续区间的下一个位置
- **输入的数字不在 Cache 中，且 Cache 已满：** 遍历 Cache 数组，找不到这个数字，且已经占用的 Cache 达到了 C，运用题目中说的 LRU 算法找到时间戳最小的位置（本质上是一个查找数组中最小元素及其位置的问题，可以参考 `E1 - shtog 挑武器`），把新的数字放进去，并更新时间戳

需要注意的是由于输入是 `int` 范围，有同学直接把读入的数字和代表 Cache 的数组中的数字相比较，然而已经初始化的数组默认的初值也是 0，这样就会造成误判；针对这个问题有以下几种解决办法：

- **把数组初值赋成一个不在 `int` 范围内的数字：** 比如超过 `int` 的数字，或者一个浮点数
- **用一个数组表示每一位是否已经被使用了：** 比如数组元素 vis_i 代表 Cache 的第 i 位是否被使用
- **用一个整数表示 Cache 数组被用到哪里了：** 不难发现任何时候 Cache 被占用的元素都是从头开始连续的一段，比如用 3 表示 Cache 的前三个位置已经有元素了

下面的题解中采用的是第二种方法。

示例代码

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

int cache[105];
int last[105];
int vis[105];

int main() {
    int c, n;
    scanf("%d%d", &c, &n);
    for(int i = 1; i <= n ;i++) {
        int x;
        scanf("%d", &x);
        int flag = 0;
        for(int j = 1; j <= c ;j++) {
            // 与输入数字相同且存过东西
            if(cache[j] == x && vis[j]) {
                printf("Cache hits, the number's position is %d.\n", j);
                // 更新时间
                last[j] = i;
                flag = 1;
            }
            if(flag)
                break;
        }

        if(flag)
            continue;

        for(int j = 1; j <= c ;j++) {
            // 没找到且 Cache 未满
            if(!vis[j]) {
                printf("Cache miss, the number's new position is %d.\n", j);
                vis[j] = 1;
                last[j] = i;
                cache[j] = x;
                flag = 1;
            }
            if(flag)
                break;
        }

        if(flag)
            continue;

        int m = 100000, pos = -1;
        // 寻找最小元素和它的位置
        for(int j = 1; j <= c ;j++)
            if(last[j] < m)
                m = last[j], pos = j;

        printf("Cache miss, the number's new position is %d.\n", pos);
        cache[pos] = x;
        last[pos] = i;
    }
    return 0;
```

```
}
```

F 小懒獭与小数计算

难度	考点
4	循环、数组

题目分析

直接利用 `double` 格式化小数点后 n 位精度远远不能满足题目的要求；我们可以想想如果没有计算机我们人是如何进行小数除法计算的：列竖式除法。

我们模拟竖式除法的过程：

我们将整数 a 除以 b，得到一个商 n 和余数 $a \bmod b$。接下来，通过逐步计算余数并将其扩大 10 倍来获得小数位数，同时进行四舍五入；逐位计算小数位数的过程：

- 使用数组 `m[]` 来保存小数部分的各位数字。
- 逐位计算每一位小数，即不断将余数 `a` 乘以 10，并除以 `b` 获得这一位的小数部分，同时更新余数为 `a * 10 % b`。
- 重复上述操作 `c + 1` 次，获得指定精度的小数位数（多一位用于下面的四舍五入）。

在检查第 $c+1$ 位是不是大于等于 5 后，如果是的话需要进位；这里要考虑的特殊情况是连续进位，甚至进位到整数部分（但不一定会进位到整数部分）

示例代码

```c
#include <stdio.h>

// m 存储小数部分
int n, m[1100];

int main() {
    int a, b, c, i;
    scanf("%d%d%d", &a, &b, &c);

    n = a / b;
    a = a % b;
    m[0] = 0;

    // 计算小数部分的 c 位
    for (i = 1; i <= c; i++) {
        m[i] = a * 10 / b;
        a = a * 10 % b;
    }

    // 判断第 c+1 位，决定是否四舍五入
    if (a * 10 / b >= 5) {
        for (i = c; i >= 0; i--) {
            m[i] = m[i] + 1; // 四舍五入进位
            if (m[i] < 10)   // 检查是否需要进一步进位
```

```
                break;
            else
                m[i] -= 10;
        }
    }

    // 如果 m[0] 有进位，就加到整数部分
    n = n + m[0];

    printf("%d.", n);
    for (i = 1; i <= c; i++) {
        printf("%d", m[i]);
    }
    return 0;
}
```

扩展延伸

`printf("%.3lf", f);` 的效果不一定是四舍五入输出到小数点后三位，这与具体的实现有关：

The rounding mode for floating-point addition is characterized by the implementation-defined value of **FLT_ROUNDS**:[18)]

-1	indeterminable
0	toward zero
1	to nearest
2	toward positive infinity
3	toward negative infinity

All other values for **FLT_ROUNDS** characterize implementation-defined rounding behavior.

在我们的 OJ 平台上的实现是四舍五入的。

G 摩卡与水獭小团体

难度	考点
5	数组、双指针法、哈希思想

题目分析

显然，对应指定的右端点 `r` ，有：

- 能与 `r` 一起构成合法的区间的左端点 `l` 的集合，一定是 `r` 之前连续的一段，即 `i`、`i + 1`，`···`，`r`
- 随着 `r` 增大，与 `r` 对应的一系列左端点中最小的一个一定不变或增加

我们使用两个整数 `l` 和 `r`，其中 `r` 代表无重复元素的子数组的右端点，`l` 代表与右端点对应的左端点中最小的一个，显然对于一对 `l` 和 `r`，它们代表的合法区间有 `r - l + 1` 个。

我们从左到右遍历数组，用下标 `r` 表示每次选取的右端点：

- 如果 `a[r]` 在当前窗口 `[l, r-1]` 中已经出现过，则移动左下标 `l` 到最近一次出现 `a[r]` 的下一个位置，这样就保证了 `[l, r]` 的子数组不包含重复元素。如何判断 `a[r]` 是否在窗口中出现过呢？可以维护一个数组 `pre`，`pre[i]` 代表 `i` 上一次出现的位置，我们只需要判断这个位置是不是在当前窗口中就可以了。
- 更新 `pre[a[r]]` 为当前元素的位置 `r`，表示元素 `a[r]` 最近一次出现的位置。

因为 `l` 和 `r` 都是单调递增的，每个元素最多会被访问两次，因此时间复杂度为 `O(n)`，可以满足题目要求。

示例代码

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

int a[500005];
int pre[500005];

int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n ;i++)
        scanf("%d", &a[i]);

    long long ans = 0;
    int l = 0;
    for(int r = 1; r <= n ;r++) {
        if(pre[a[r]] >= l)
            l = pre[a[r]] + 1;    // 如果a[r]在[l, r-1]之间出现过，更新左边界
        pre[a[r]] = r;             // 更新a[r]最后出现的位置
        ans += r - l + 1;          // 当前窗口长度为无重复元素子数组数量
    }

    // 别忘了分析数据范围
    printf("%lld", ans);
    return 0;
}
```

扩展延伸

双指针法

H ddz 的强迫症

难度	考点
6	前缀和、循环

题目分析

如果我们最终能得到 m 个相同的数 $[x, x, x, \ldots, x]$，我们在最左边的三个数中间插入异或，由于 $x \oplus x \oplus x = x$，我们会得到 $m - 2$ 个相同的数 $[x, \ldots, x]$，重复此操作最终我们可以得到不超过三个相同的数。也就是说我们只需要判断原数组是否能够切割成两份或者三份，每一份异或的结果相同。

对于两份的情况，如果我们最终得到两个相同的数 $[x, x]$，那么他们的异或是 0，也即原数组所有数异或的结果是 0。充分性显然，所以原数组所有数异或的结果是 0 就输出 `Yes`。

对于三份的情况，由于 n 不大，我们可以枚举分割的位置，对于每种分割方法计算每一份的异或是否相同。为了减少计算次数，计算每一份的异或的时候可以用一个类似前缀和的数组来快速计算。

示例代码

```c
#include <stdio.h>

int pre[2005];

int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        int n;
        scanf("%d", &n);
        for (int i = 1; i <= n; ++i) {
            scanf("%d", &pre[i]);
            pre[i] ^= pre[i - 1];
        }
        if (pre[n] == 0) {
            printf("Yes\n");
            continue;
        }
        int yes = 0;
        for (int i = 1; i < n && !yes; ++i) {
            for (int j = i + 1; j <= n && !yes; ++j) {
                // 判断是否有 pre[i] = pre[j] ^ pre[i] = pre[n] ^ pre[j]
                // 化简上述式子就能得到
                if (pre[j] == 0 && pre[n] == pre[i]) {
                    printf("Yes\n");
                    yes = 1;
                }
            }
        }
        if (!yes) {
            printf("No\n");
        }
    }
    return 0;
}
```

I ddz 与 vya 的游戏

难度	考点
6	游戏、质数

题目分析

1. $n=1$ 时无法操作所以先手输。
2. 如果 n 是 2 的幂次，由于 n 没有奇因数，先手只能减一。

- 如果 $n=2$，后手得到 1，后手输。
- 如果 $n>2$，后手得到一个奇数，后手除以这个奇数，先手得到 1，先手输。

3. 其他情况我们考虑对 n 进行质因数分解 $n=2^a3^b5^c\cdots$

- $a=0$ 时也就是 n 为奇数时，先手除以 n 本身，后手得到 1 后手输。
- $a=1$ 时
  - 如果除 2 以外只有一个质数，先手如果减一，后手拿到奇数，后手除以这个奇数，先手拿到 1，先手输。先手如果除以那个质数，后手拿到 2，后手减一，先手输。所以此时先手必输。
  - 如果除 2 以外多于一个质数，$n=2pqr\cdots$，先手除以 $2p$ 以外的其他质数的积使得 后手拿到 $2p$，此时套用前面的讨论结果可知此时后手必输
- $a>1$ 时，先手除以除 2 以外所有质因数的积，后手拿到 $2^a(a>1)$，根据前面 2 的幂次的讨论，后手输。

综上所述是当 $n=1,2^x(x>1),2p(p>2且p是质数)$ 时后手 vya 必胜其余先手必胜。

示例代码

```
#include<stdio.h>

int isPrime(int n) {
    if (n == 1) {
        return 0;
    }
    for (int i = 2; i * i <= n; ++i) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}

int main(){
    int t, n;
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n);
        if (n == 1 || (n > 2 && (n & (n - 1)) == 0) || (n % 2 == 0 && ((n / 2) &
1) && isPrime(n / 2))) {
            printf("Yes\n");
        } else {
```

```
            printf("No\n");
        }
    }
    return 0;
}
```

J 虾泥卡拉斯

难度	考点
7	模拟、代码能力

题目分析

耐心读题，谨慎写题。

几个需要注意的地方：

1. base表可以找些等差等比的规律推出来，可能会比纯手搓稍微方便些
2. 每周前记得把 sp 花完，把上限都点了。

详见代码注释。

示例代码

```
#include<stdio.h>
int c[10]={0,0,0,0,0,0,1},lim[10]={0,500,500,500,500};    //培育偶像的属性值和属性上限
int b[10],p[10];      //支援偶像的羁绊值和每周的位置
int num[10],B[10],lv[10]={0,1,1,1,1,1,1},cnt[10];    //房间的人数、羁绊值总和、等级、上课次数
int VO[10],DA[10],VI[10],ME[10];
int AUDI[10]={0,1000,10000,50000,100000},GOAL[10]={0,1000,10000,100000,500000};
//试镜所需数值、试镜成功的粉丝数、目标人数
char T[10];           //每周的指令
int base[10][10][10]={
    {
        {20,0,0,0,3,32},
        {0,20,0,0,3,32},
        {0,0,20,0,3,32},
        {3,0,0,15,3,158},
        {0,3,0,3,10,158},
        {0,0,5,5,5,263}
    }
};
int main(){
    for(int i=1;i<5;i++){
        for(int j=0;j<6;j++)for(int k=0;k<6;k++)base[i][j][k]=base[i-1][j][k];
        base[i][0][0]+=4,base[i][0][4]+=1,base[i][0][5]*=2;
        base[i][1][1]+=4,base[i][1][4]+=1,base[i][1][5]*=2;
        base[i][2][2]+=4,base[i][2][4]+=1,base[i][2][5]*=2;
        base[i][3][0]+=1,base[i][3][3]+=5,base[i][3][4]+=1,base[i][3][5]*=2;
        base[i][4][1]+=1,base[i][4][3]+=1,base[i][4][4]+=2,base[i][4][5]*=2;
```

```
        base[i][5][2]+=1,base[i][5][3]+=1,base[i][5][4]+=1,base[i][5][5]*=2;
    }//生成base表
    for(int i=1;i<=5;i++)scanf("%d",&c[i]);
    for(int i=1;i<=5;i++)scanf("%d",&b[i]);
    for(int i=1;i<=4;i++)scanf("%d%d%d%d",&VO[i],&DA[i],&VI[i],&ME[i]);
    for(int season=1;season<=4;season++){
        for(int week=1;week<=8;week++){
            for(int i=1;i<=3;i++)lim[i]+=c[5]/3;
            c[5]%=3;//每周前用sp把上限点了
            for(int i=1;i<=5;i++)scanf("%d",&p[i]);
            scanf("%s",T);
            if(T[0]=='l'){
                int pos=0,res=0;
                for(int i=1;i<=6;i++)num[i]=0,B[i]=0;
                for(int i=1;i<=5;i++)num[p[i]]++,B[p[i]]+=b[i]; //计算每个房间的人
数、总羁绊值
                for(int i=1;i<=6;i++)if(res<num[i])res=num[i],pos=i; //找到人数最
多、编号最小的房间
                for(int i=1;i<=6;i++){
                    c[i]=c[i]+base[lv[pos]-1][pos-1][i-1]*(1+0.05*res)*
(1+0.001*B[pos]);
                    if(i<=4&&c[i]>lim[i])c[i]=lim[i]; //不能超上限
                }
                cnt[pos]++;
                if(cnt[pos]==4){
                    lv[pos]++,cnt[pos]=0;
                    if(lv[pos]>5)lv[pos]=5;
                }
                //上课次数计数和升房
                for(int i=1;i<=5;i++)if(p[i]==pos){
                    b[i]+=5;
                    if(b[i]>100)b[i]=100;
                }
                //支援偶像羁绊值增加，注意100的上限
            }
            else
if(c[1]>=VO[season]&&c[2]>=DA[season]&&c[3]>=VI[season]&&c[4]>=ME[season])
                c[6]+=AUDI[season];
                //试镜
        }
        if(c[6]<GOAL[season]){
            printf("%d\n",season);
            for(int i=1;i<=4;i++)printf("%d ",c[i]);
            return 0;
            //季度结算
        }
    }
    puts("True end");
    for(int i=1;i<=4;i++)printf("%d ",c[i]);
}
```