A.圣剑plus

```c
#include <stdio.h>

int n, m, t, num = 0; // n为输入的数字个数，m为当前有效数字个数，t用于交换，num用于记录总和
int a[2005]; // 数组a用于存储输入的数字

int main()
{
    scanf("%d", &n); // 读取数字个数n
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]); // 读取n个数字到数组a中

    m = n; // 初始化有效数字个数m为n

    for (int k = 0; k < n - 1; k++) { // 进行n-1次操作
        // 冒泡排序，按降序排列数组a
        for (int i = 0; i < m; i++)
            for (int j = i + 1; j < m; j++)
                if (a[i] < a[j]) // 如果a[i]小于a[j]，则交换
                    t = a[i], a[i] = a[j], a[j] = t;

        // 将当前最大两个数的和加入到num中
        num += a[m - 1] + a[m - 2];
        // 更新第二大的数
        a[m - 2] += a[m - 1];
        m--; // 有效数字个数减1
    }

    // 每次都挑选重量最小的两个碎片进行修复
    printf("%d", num); // 输出最终的总和
}
```

B.小球反弹

[原题链接]

> 题解:
> 在长宽上暴力找两个常数x, y;使得 x * a : y * b == c : d;
> 进而求得该矩形对角线的距离，由于小球要回到原点(即左上角)，因此，对角线*2即为答案

> 题解一:

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

int main()
{
    int a, b, c, d; // 声明四个整数变量a, b, c, d
    // 读取四个整数的值
```

```
    scanf("%d %d %d %d", &a, &b, &c, &d);

    long long t = 1; // 初始化t为1，用于找到合适的倍数
    while (1) // 无限循环，直到找到合适的t
    {
        // 检查(c * t)是否能被a整除，且(d * t)是否能被b整除
        if ((c * t) % a == 0 && (d * t) % b == 0)
            break; // 如果满足条件，跳出循环
        t++; // 如果不满足条件，t加1
    }

    // 计算并输出结果，格式为浮点数保留两位小数
    printf("%.2f", 2 * sqrt(c * c * t * t + d * d * t * t));
    return 0; // 程序正常结束
}
```

> 题解二:

```
#include <bits/stdc++.h> // 包含常用的C++库

using namespace std;

int main()
{
    int a, b, c, d; // 声明四个整数变量a, b, c, d
    cin >> a >> b >> c >> d; // 读取四个整数的值
    int A = a, B = b; // 备份初始的a和b的值，用于后续计算

    // 当(a/b)与(c/d)的差绝对值大于1e-6时，继续循环
    while (abs((double)a / b - (double)c / d) > 1e-6)
    {
        // 如果(a/b)大于(c/d)，增加b的值
        if ((double)a / b > (double)c / d)
            b += B; // 增加b的初始值
        else
            a += A; // 否则增加a的初始值
    }

    // 计算并输出结果，格式为浮点数保留两位小数
    printf("%.2lf", 2 * sqrt((double)a * a + b * b));
    return 0; // 程序正常结束
}
```

C. 虚晃一枪

```c
#include <stdio.h>

int main()
{
    printf("Yzhgg把%%100的温柔都给了Zmzjj\n");

    return 0;
}
```

D.传送门

卡特兰数（进阶做法）

```c
#include <stdio.h>

// 定义递归函数dg，用于计算可能的队伍组合
int dg(int m, int n) {
    // 如果人类数量大于精灵数量，或者人类数量为0，返回0
    if (n > m || m == 0) return 0;
    // 如果队伍中没有人类，情况成立，返回1
    else if (n == 0) return 1;
    // 否则，递归计算可能的组合情况
    else return dg(m - 1, n) + dg(m, n - 1);
}

int main() {
    int n, m; // 声明人类数量n和精灵数量m
    // 读取人类和精灵的数量
    scanf("%d %d", &m, &n);
    // 调用递归函数dg，并输出结果
    printf("%d", dg(m, n));
    return 0; // 程序正常结束
}
```

E.一元一次方程

```c
#include<stdio.h> // 引入标准输入输出库
#include<string.h> // 引入字符串处理库

char cc[105], c, a; // cc用于读取方程式，c为当前字符，a为未知数名称
long long int f = 1, now = 1, k, b, x, r; // f用于记录符号，now为方程左边的一元一次部分，k为未知数系数和，b为常数项和，x为当前常数项，r用于判断是否有数字读入

int main() {
    scanf("%s", cc); // 读取方程式字符串
```

```
    for (int i = 0; i < strlen(cc); i++) { // 遍历字符串中的每个字符
        c = cc[i]; // 当前字符

        // 处理负号
        if (c == '-') {
            b += now * f * x; // 将当前常数项累加到b中
            x = 0; // 重置当前常数项
            f = -1; // 更新符号为负
            r = 0; // 重置数字读入标志
        }

        // 处理加号
        if (c == '+') {
            b += now * f * x; // 将当前常数项累加到b中
            x = 0; // 重置当前常数项
            f = 1; // 更新符号为正
            r = 0; // 重置数字读入标志
        }

        // 处理等号
        if (c == '=') {
            b += now * f * x; // 将当前常数项累加到b中
            x = 0; // 重置当前常数项
            f = 1; // 符号重置为正
            now = -1; // 更新now为负以处理等号右边
            r = 0; // 重置数字读入标志
        }

        // 处理未知数
        if (c >= 'a' && c <= 'z') {
            if (r) {
                k += now * f * x; // 如果前面有数字，累加系数
                x = 0; // 重置当前常数项
            } else {
                k += now * f; // 否则只累加符号
            }
            a = c; // 记录未知数名称
            r = 0; // 重置数字读入标志
        }

        // 处理数字
        if (c >= '0' && c <= '9') {
            x = x * 10 + c - '0'; // 计算当前常数项
            r = 1; // 设置数字读入标志
        }
    }

    b += now * f * x; // 加上最后一项常数（如果最后一项是未知数，则会加0）

    double ans = (double)(-b * 1.0 / k); // 计算未知数的值
    if (ans == -0.0) ans = 0; // 特判，将-0.0改为0

    // 输出结果，保留三位小数
    printf("%c=%.3lf\n", a, ans);
    return 0; // 程序正常结束
}
```

F.A - B

原题链接

> 题解：A - B = C; A = B + C
> 先记录 A 出现的次数， 然后重新遍历一遍求 B + C 出现的次数和即可(即cnt += ant[B + C]);

```
#include <stdio.h> // 引入标准输入输出库

int ant[1000010], a[1000010]; // ant数组用于记录每个数出现的次数，a数组用于存储输入的数

int main()
{
    int n, c; // n为输入的数的数量，c为差值
    scanf("%d %d", &n, &c); // 读取n和c的值

    // 读取n个数并记录每个数出现的次数
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &a[i]); // 读取每个数
        ant[a[i]]++; // 增加该数的出现次数
    }

    long long cnt = 0; // 初始化计数器cnt为0

    // 遍历数组a，计算满足条件的数对
    for (int i = 0; i < n; ++i)
    {
        cnt += ant[a[i] + c]; // 统计与当前数相差c的数的出现次数
    }

    printf("%lld", cnt); // 输出符合条件的数对的总数
    return 0; // 程序正常结束
}
```

G.卡片对抗赛

```
#include<bits/stdc++.h> // 引入常用的C++库
using namespace std;
#define int long long // 将int类型定义为long long，方便处理大数

const int N = 2e5 + 10; // 定义常量N，作为数组的最大大小

signed main() {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0); // 优化输入输出
```

```
    int T; // 测试用例数量
    cin >> T; // 读取测试用例数量
    while (T--) { // 遍历每个测试用例
        int n, k; // n为数组大小，k为额外的值
        cin >> n >> k; // 读取n和k的值
        int mx = 0, sum = 0; // mx用于存储数组中的最大值，sum用于存储数组元素的总和
        for (int i = 0, num; i < n; ++i) { // 遍历n个数
            cin >> num; // 读取当前数
            sum += num; // 累加总和
            mx = max(mx, num); // 更新最大值
        }
        // 计算v的初步值，限制为n和(sum + k) / mx中的较小值
        int v = min(n, (sum + k) / mx);
        // 确保(v * (v + 1) / 2)不超过sum + k
        while ((sum + v - 1) / v * v > sum + k)
            v--; // 如果条件不满足，减少v的值
        cout << v << '\n'; // 输出结果
    }
}
```

数字国度（easy）

```
#include<bits/stdc++.h>
using namespace std;

int main() {
    for (int i = 0; i <= 31; i++) {
        cout << (1LL << i) << ' ';
    }
}
```

```
1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536 131072 262144
524288 1048576 2097152 4194304 8388608 16777216 33554432 67108864 134217728
268435456 536870912 1073741824 2147483648
```

I.完美字符串

原题链接

> 题解：
> 题目比较绕，题目本质就是找一个字符串中出现最多的字母的次数，长度 - 该次数 即为答案

```c
#include <stdio.h> // 引入标准输入输出库

int a[30]; // 用于记录每个字符出现的次数，假设只处理小写字母

int main() {
    char cp; // 当前读取的字符
    int n; // 字符的数量

    scanf("%d", &n); // 读取字符的数量
    int maxc = 0; // 记录出现次数最多的字符的数量
    int t = n; // 变量t用于循环控制
    getchar(); // 读取换行符以清除输入缓冲区

    // 循环读取n个字符
    while(t--) {
        char cp = 'a'; // 初始化当前字符
        scanf("%c", &cp); // 读取一个字符
        a[cp - 'a']++; // 统计字符cp出现的次数
        // 更新最大出现次数
        if(maxc < a[cp - 'a']) {
            maxc = a[cp - 'a'];
        }
    }

    // 输出n减去出现次数最多的字符的数量
    printf("%d", n - maxc);
    return 0; // 程序正常结束
}
```