骗分拿省一教程

第一招：判断性题，直接输出固定结果。

阅读题目抓住关键点：

如果……情况，纯输出计算出的……答案，如果……，则输出false。

想必大家都看到关键词了，”false“。那么我们直接骗分。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    cout<<"false"<<endl;
    return 0;
}
```

这样就可以骗到50分了。

但要注意，如果出现下述这种情况，就不能这样骗分！！！

……多组数据输入，输出每行为……答案。

```
输入样例：
    6 //共6组数据输入
    1
    2
    3
    4
    5
    6
输出样例：
    20
    false
    1
    false
    false
    114514
```

千万注意，这种不能这样骗分！！！

第二招：特殊数据直接输出

Alice 和 Bob 打算给花园里的 n 株植物浇水。植物排成一行，从左到右进行标记， 编号从 0 到 n - 1 。其中，第 i 株植物的位置是 x = i 。 每一株植物都需要浇特定量的水。Alice 和 Bob 每人有一个水罐，最初是满的 。 他们按下面描述的方式完成浇水： Alice 按 从左到右 的顺序给植物浇水，从植物 0 开始。Bob 按 从右到 左 的顺序给植物浇水，从植物 n - 1 开始。他们 同时 给植物浇水。 无论需要多少水，为每株植物浇水所需的时间都是相同的。 如果 Alice/Bob 水罐中的水足以 完全 灌溉植物，他们 必须 给植物浇水。 否则，他们 首先（立即）重新装满罐子，然后给植物浇水。 如果 Alice 和 Bob 到达同一株植物，那么当前水罐中水 更多 的人会给这 株植物浇水。如果他俩水量相同，那么 Alice 会给这株植物浇水。 给你一个下标从 0 开始的整数数组 plants ，数组由 n 个整数组成。其中， plants[i] 为第 i 株植物需要的水

量。另有两个整数 capacityA 和 capacityB 分别表示 Alice 和 Bob 水罐的容量。返回两人浇灌所有植物过程中重新灌满水罐的 次数 。 输入第一行为三个数，分别为植物数n，capacityA和capacityB 第二行为n个植物需要的水量，每两个数之间以空格分隔。 第二行为两人总共需要的重新灌水次数。 示例 1： 1 2 18 3 19 6 5 4 示例 2： 示例 3： 提示： 1 <= n <= 10 5 1 <= plants[i] <= 10 6 max(plants[i]) <= capacityA, capacityB <= 10 9

输入：plants = [2,2,3,3], capacityA = 5, capacityB = 5 输出：1 解释： - 最初，Alice 和 Bob 的水罐中各有 5 单元水。 - Alice 给植物 0 浇水，Bob 给植物 3 浇水。 - Alice 和 Bob 现在分别剩下 3 单元和 2 单元水。 - Alice 有足够的水给植物 1 ，所以她直接浇水。Bob 的水不够给植物 2 ，所 以他先重新装满水，再浇水。 所以，两人浇灌所有植物过程中重新灌满水罐的次数 = 0 + 0 + 1 + 0 = 1 。

输入：plants = [2,2,3,3], capacityA = 3, capacityB = 4 输出：2 解释： - 最初，Alice 的水罐中有 3 单元水，Bob 的水罐中有 4 单元水。 - Alice 给植物 0 浇水，Bob 给植物 3 浇水。 - Alice 和 Bob 现在都只有 1 单元水，并分别需要给植物 1 和植物 2 浇 水。 - 由于他们的水量均不足以浇水，所以他们重新灌满水罐再进行浇水。 所以，两人浇灌所有植物过程中重新灌满水罐的次数 = 0 + 1 + 1 + 0 = 2 。

输入：plants = [5], capacityA = 10, capacityB = 8 输出：0 解释： - 只有一株植物 - Alice 的水罐有 10 单元水，Bob 的水罐有 8 单元水。因此 Alice 的水罐 中水更多，她会给这株植物浇水。 所以，两人浇灌所有植物过程中重新灌满水罐的次数 = 0

大家可以仔细看一下特殊的输出数据是什么。

揭晓答案：”两人浇灌所有植物过程中重新灌满水罐的次数 = 0“ 中的 “0” 。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    cout<<0<<endl;
    return 0;
}
```

这个代码是我在2024年10月5日也就是十一期间复赛第一天集训的时候，下午最后一道题的骗分记录。最终得分20分，也挺不错的了。

第三招：打表测试点，又称：刷测试点

这个方法特别常用，属于是考场骗分的必备技能了，常用于题干可以看懂可以理解可以会做，但是落在代码上却不会实现的情况。

方法就是列举多组输入数据，并手算其结果，数据越多，验算次数越多，越精准，越有概率抽到测试点。

```
if(n==1) cout<<1<<endl;
if(n==2) cout<<2<<endl;
if(n==3) cout<<5<<endl;
if(n==4) cout<<6<<endl;
if(5<=n&&n<10) cout<<114514<<endl;
if(11<=n&&n<20) cout<<444<<endl;
if(21<=n&&n<25) cout<<60<<endl;
else cout<<-1<<endl;
```

但这种方法仅仅适用于题干简单，易于理解，一次输入数据个数少（1-2个）。

但如果这道题是一个数学问题，那么有可能在刷测试点的过程中发现某些规律，虽然不确定是否正确，但既然规律发现了，就应该遵循，尝试模拟这个规律，这样骗来的分甚至有可能会50+。

第四招：拿我能拿到的分

正常复赛的试卷上会有写本题测试点的数据范围分布，如XXX个测试点范围为1<=n<=100，XXXX个测试点范围为……，我们可能无法考虑所有测试点的骗分，那就考虑我们能够拿到的一些测试点的AC。针对数据范围的骗分几个思路。

1.暴力枚举，正常按照样例规范，暴力枚举不一定能拿到高分，但起码只要枚举是正确的，就一定有分可以得到。例如2024年10月5日下午第二题，是一个整数查重的题，我使用了2层循环，因为最大的样例范围是5x10^4，所以最终70分到手，如果样例再大一些，可能我也就四五十分吧。

2.卡极限数值。例如求峰值数量的题（峰值即为a[i-1]<a[i]且a[i]>a[i+1]，当然这个例子不太恰当，因为明显有更好的解决办法），但是在处理最后一个数的时候，我可以给他开成特别大的极限数值，然后保证我程序的计算无误。当然这个例子不恰当，很明显有更好的解决办法，但这个思想是可以学习一下的。

3.打表直接打到某个范围，当然除非题干给的相当友好，否则真的没必要这样做。

同时除了针对数据范围的骗分，还有针对其他各种方面的骗分，具体就不一一列举了。

第五招：从出题人的角度来理解问题，进行骗分。

最明显的就是前面提到的直接输出特殊值，同时我们还可以考虑到特殊值之下的各种情况，考虑对应的骗分策略即可。

第六招：重点关注样例和样例解释

从正常做题的角度来说，我们关注样例和样例解释的目的是更好更全面更准确地理解题目意思。

从骗分角度来说，样例解释为辅助，样例内容有可能会提到部分特殊值和结果的情况，这些都是可以启示我们骗更多分的。

但要记住，虽然历年复赛没有测试点包含样例的情况，但CCF官方也没有正式的对此声明过，对于一些样例中的特殊值情况，测试点出现的概率仍是极大的。针对题目不同条件的骗分，也应从样例入手，有可能会对骗分有好处。