Missing

题目描述

编号为 1 到 100 100 的牛中消失了 100 头，剩下的对半分给 Alice 和 Bob，两人每人各得到 50 000 头。现在 Alice 只能向 Bob 发送 100 比特，Bob 需要根据 Alice 提供的信息找出任意一头消失的牛的编号。

任务要求

你需要实现两个函数 `farm1` 和 `farm2`，签名如下：

```
void farm1(char *msg, int *cows);
int farm2(char *msg, int *cows);
```

`farm1` 表示 Alice 的操作。其中

- `cows` 指向一长度为 50 000 的 `int` 数组，你可以从中读取 Alice 得到的牛的编号；
- `msg` 指向一长度为 100 的 `char` 数组。你需要在其中每一个位置均填入 `'0'` 或 `'1'`，表示 Alice 向 Bob 发送的 100 比特。

`farm2` 表示 Bob 的操作。其中

- `cows` 和 `farm1` 的参数类似，亦指向一长度为 50 000 的 `int` 数组，你可以从中读取 Bob 得到的牛的编号；
- `msg` 指向一长度为 100 的 `char` 数组，其中每个字符均为 `'0'` 或 `'1'`，表示 Alice 向 Bob 发送的 100 比特；
- 函数应返回一 `int`，表示 Bob 找出的消失的牛的编号。

本地测试

下发文件中提供一供本地测试的 `missing_driver.cpp`。你需将它和你的答案文件一同编译以生成可执行文件来本地测试。例如，若你的答案文件叫 `missing.cpp`，可通过

```
g++ missing_driver.cpp missing.cpp -o missing
```

来生成可执行文件 `missing`。生成的可执行文件会从标准输入读取 100 000 个互不相同的 1 到 100 100 的数字，前 50 000 个为 Alice 得到的牛的编号，后 50 000 个为 Bob 得到的牛的编号。它会调用你的 `farm1` 和 `farm2`，并根据情况输出相应信息。特别地，如果输出 `OK` 表示本地测试正确。

注意：在线测试时用全局变量等手段在 Alice 和 Bob 间悄悄通信不会有效。但你仍然可以正常使用全局变量。

样例

下给出一个简化的样例，其中开始时有 10 头牛，消失了 2 头，Alice 和 Bob 各得到所剩 8 头牛中的 4 头。你可以将 `missing_driver.cpp` 中的常量 `N` 改为 10，`N1` 和 `N2` 改为 4 来使用此样例。

输入

```
1
3
5
7
2
4
6
8
```

解释

此时，Alice 会得到 1、3、5、7 号牛，Bob 则得到 2、4、6、8 号，剩余的 9 号和 10 号消失了。在程序运行中，

1. `farm1` 会被调用一次，参数 `msg` 为一待填入的缓冲区，`cows` 指向长度为 4 的 `int` 数组，内容为 $[1, 3, 5, 7]$；
2. 如果 `farm1` 正确执行，`farm2` 也会被调用一次，参数 `msg` 为 `farm1` 之前写入的比特字符串，`cows` 指向长度为 4 的 `int` 数组，内容为 $[2, 4, 6, 8]$；如果 `farm2` 返回 9 或 10，则此通过此样例。

下发文件

- `missing_driver.cpp` 意义如上。
- `missing.cpp` 提供了一个样例答案，Alice 会根据是否得到 1 号牛输出 100 个 `'0'` 或 `'1'`，Bob 则根据此信息，在确定 1 号牛丢失时输出 1，否则输出 2。