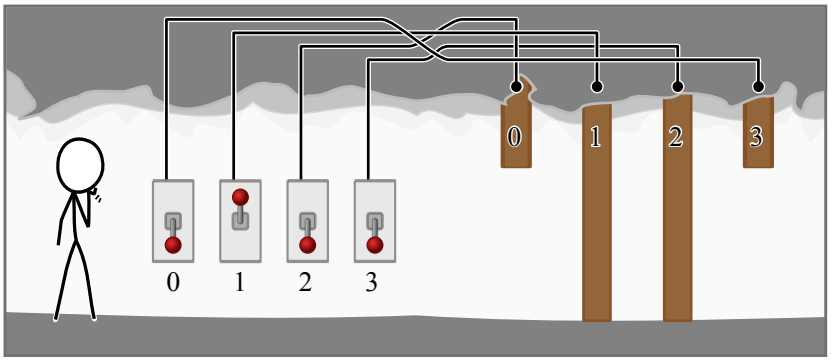
International Olympiad in Informatics
2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

中文(澳門) — 1.0

在你迷失在前往UQ中心的路途上時，你無意中發現一個進入大學地底秘密洞穴的入口。這個入口被 `N` 扇連續的門所阻擋著，每扇門是在前一扇門的後面。同時有 `N` 個開關，每一個開關連接著一扇門。



門的編號順序是 `0, 1, ⋯, (N - 1)`，其中編號 0 的一扇門是最近你的一扇門。開關的編號亦是 `0, 1, ⋯, (N - 1)`，但是你並不知道哪個開關是連接到哪扇門的。

所有開關都位於洞穴的入口處。每個開關可以處於 上 或 下 兩個位置。每個開關只有一個位置是該開關的正確位置。若一個開關處於正確位置時，與之相連的那一扇門就會是開著的，反之，若一個開關並不是處於正確位置時，與之相連的那一扇門就會是關閉著。每個開關的正確位置可以是不同的，同時你亦不知道哪一個位置才是正確位置。

你想要了解這個秘密的開關系統。為了達到這個，你可以把這些開關設置成不同位置的組合，然後走進洞穴裏看看哪一扇門是第一扇關閉著的門。這些門並不是透明的，因此當你遇上第一扇關閉著的門時，你就不能看到在那扇門以後的門的狀態。

你有時間可以試不多於 `70,000` 個不同的開關位置組合。你的任務是嘗試決定每一個開關的正確位置，以及決定哪一個開關是連到哪一扇門的。

程式實現

你需要提交一個程式檔案， 其內將包含子程式 `exploreCave()`。 該子程式最多可以調用 grader 的函數 `tryCombination()` 70,000 次， 然後必須要調用另一個 grader 的子程式 `answer()` 作為程式的完結。這些子程式的詳情請參閱下面說明。

Grader 函數: **tryCombination()**

C/C++
```
int tryCombination(int S[]);
```

Pascal
```
function tryCombination(var S: array of LongInt) : LongInt;
```

描述

grader 將為你提供這個函數。它容許你嘗試一個開關的組合， 並進入洞穴找出第一扇關閉著的門。 若所有門都是開著的， 這函數的返回值將個是 `-1`。 這個函數的執行時間為 `O(N)`； 即是說在最差的情況下， 這函數的執行時間是與 `N` 成正比。

你可以調用這函數最多 `70,000` 次。

參數

- `S`: 一個長度為 `N` 的陣列， 它指明每個開關的位置。 陣列中元素 `S[i]` 將對應開關 `i`。 若它的值是 `0` 則表示開關是在 `上` 的位置， 而若它的值是 `1` 則表示開關是在 `下` 的位置。
- 返回值: 函數將返回第一扇關閉著的門的編號， 若所有門都是開著的話， 則本函數的返回值將會是 `-1`。

Grader 子程式: **answer()**

C/C++
```
void answer(int S[], int D[]);
```

Pascal
```
procedure answer(var S, D: array of LongInt);
```

描述

當你已經找到所有開關的正確位置並且找出哪個開關是連接哪一扇門時， 你需要調用此子程式。

參數

- `S`：一個長度為 `N` 的陣列，它指出每個開關的正確位置。它的格式和上述的 `tryCombination()` 內所描述的格式相同。
- `D`：一個長度為 `N` 的陣列，它指出每個開關是和哪扇門相連的。具體來說，元素 `D[i]` 應含有與開關 `i` 相連的那扇門的編號。
- 返回值：這個子程式不會返回，它會結束整個程式的運行。

你要編寫的子程式：**`exploreCave()`**

C/C++
```
void exploreCave(int N);
```

Pascal
```
procedure exploreCave(N: longint);
```

描述

你提交的檔案必須要實現這個子程式。

這個函數必須調用 grader 子程式 `tryCombination()` 以決定每個開關的正確位置及開關與門的相連關係，且當你有了答案後，你必須調用 `answer()`。

參數

- `N`：開關及門的數目。

樣例程序

假設開關及門的組合如圖所示：

函數調用	返回值	解釋
`tryCombination([1, 0, 1, 1])`	`1`	這是對應於上圖的。開關 0, 2 及 3 是在 `下` 的位置，而開關 1 是在 `上` 的位置。函數返回值是 `1`，它表示編號為 1 的門是第一扇關閉著的門（由左邊計起）。
`tryCombination([0, 1, 1, 0])`	`3`	門 0, 1 及 2 將會開著，而門 3 將會關閉著。
`tryCombination([1, 1, 1, 0])`	`-1`	將 0 號開關設為 `下` 將會使所有門都打開，而函數的返回值是 `-1`。
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	(Program exits)	我們猜想開關組合 `[1, 1, 1, 0]` 就是正確的組合，且開關 0, 1, 2 及 3 分別對應於門 3, 1, 0 及 2。

限制條件

- 時間限制: 2 秒
- 記憶體限制: 32 MB
- `1 ≤ N ≤ 5,000`

子任務

子任務	分數	附加的輸入限制
1	12	對每一個 `i`, 開關 `i` 是連接到門 `i`。 你的任務只是要決定開關的正確位置組合。
2	13	開關的正確組合將會是 `[0, 0, 0, …, 0]`。 你的工作只是要決定哪個開關是對應於哪扇門。
3	21	`N ≤ 100`
4	30	`N ≤ 2,000`
5	24	(None)

實驗

你電腦上的樣例 grader 將讀入文字檔 `cave.in`, 它的格式如下:

- 第 1 行: `N`
- 第 2 行: `S[0] S[1] … S[N - 1]`
- 第 3 行: `D[0] D[1] … D[N - 1]`

在這裡, `N` 是門及開關的數目, `S[i]` 是開關 `i` 的正確位置, 而 `D[i]` 則是開關 `i` 所連接到的門的編號。

例如: 上面例子的資料將會是用以下格式的:

```
4
1 1 1 0
3 1 0 2
```

編譯語言注意事項

C/C++	你必須包括 `#include "cave.h"`.
Pascal	你必須定義 `unit Cave` 單元，並且在單元內你必須透過 `uses GraderHelpLib` 來導入 grader 的函數。所有矩陣均由 `0` 開始 (而不是由 `1` 開始).

請參閱在你電腦上的例子題解模板。