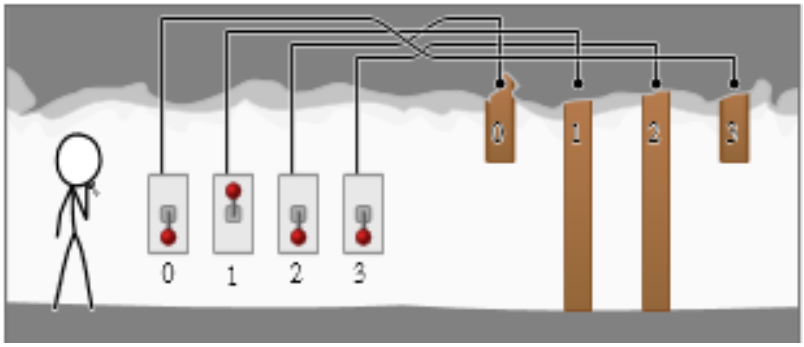
你在從學院到 UQ 中心的漫長路程中迷路了。你意外地發現了通往學校地底秘密洞穴系統的入口。這個入口被一個有連續 N 個門的安全系統阻隔。這個系統有 N 個開關，分別連接到這 N 個不同的門。



這些門依序由 0, 1, …, N–1 編號，其中最接近你的那扇門編號為 0。開關的編號也是 0, 1, …, N–1，但是你並不知道哪一個開關連到哪一扇門。

開關都位於洞穴的入口處。每個開關可以被設定成**上**或**下**的位置。每一個開關只有一個位置是正確的。若一個開關被設定在正確的位置，則其連接到的門便會打開；若一個開關被設定在不正確的位置，則其連接到的門便會關閉。每一個開關的正確設定位置可能不相同，現在你並不知道開關的正確設定位置。

你想要了解這套安全系統。為了達到這個目的，你可以將開關設定成任何的位置組合，並且走入這個洞穴觀察第一個未開的門。這些門都是不透明的，當你發現第一個未開的門時，你無法看到任何後方的門。

你可以嘗試最多 70,000 種開關的位置組合。你的任務是決定每個開關的正確設定位置，以及每個開關連接到哪一扇門。

程式實作

你必須繳交一個原始碼檔，實作 `exploreCave()` 這個函式。這個函式最多可以呼叫評分程式中的函式 `tryCombination()` 70,000 次，而且最後必須呼叫評分程式中的函式 `answer()`。這些函式的描述如下。

評分程式函式 `tryCombination()`

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

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

函式描述

評分程式會提供這個函式。這個函式允許你測驗一個開關設定組合，並得以進入洞穴來查看第一個未打開的門。假如所有的門都被打開了，這個函式會回傳 -1。這個函式的時間複雜度是 $O(N)$；也就是說，最差的情況下執行時間是和 N 成正比。

這個函式最多可以被呼叫 70,000 次。

參數說明

- *S*: 長度為 N 的陣列，表示每個開關的設定位置。元素 $S[i]$ 對應到開關 i，其值為 0 代表開關位置向上，其值為 1 代表開關位置向下。
- 回傳值：第一個未打開的門的編號。若所有的門都打開，回傳值為 -1。

評分程式函式 `answer()`

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

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

函式描述

當你已經找出能打開所有門的開關位置設定和開關與洞門的連結關係時，請呼叫這個函式。

參數說明

- *S*: 長度 N 的陣列，代表每個開關的正確設定位置。其元素代表意義請參閱 `tryCombination()` 中的 *S* 說明。
- *D*: 長度 N 的陣列，代表每個開關連結的洞門。換句話說，元素 $D[i]$ 代表開關 i 連結到的洞門編號。
- 回傳值：這個函式沒有回傳值，且會結束整個程式。

你的函式 exploreCave()

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

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

函式描述

你的解答必須實作這個函式。這個函式應該運用 tryCombination() 來決定每個開關的正確設定位置與每個開關連結的洞門，而且必須在確認答案後呼叫 answer() 一次來作答。

參數說明

- *N*: 洞內中的開關/洞門的數目。

執行範例

假設洞門和開關如同第一頁的圖片所示：

函式呼叫	回傳值	說明
`tryCombination([1, 0, 1, 1])`	`1`	參閱第一頁的圖片。開關 0, 2 和 3 的位置向下，開關 1 的位置向上。函式回傳 1，代表洞門編號 1 是第一個沒開的門。
`tryCombination([0, 1, 1, 0])`	`3`	洞門 0, 1, 2 打開了，洞門 3 未開。
`tryCombination([1, 1, 1, 0])`	`-1`	把開關 0 的位置設定向下，此時回傳 -1，代表所有的門都打開了。
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program exits)*	我們猜測開關正確設定位置是 [1, 1, 1, 0]，而且開關 0, 1, 2, 3 連結到洞門 3, 1, 0, 2。

任務限制

- 時間限制：2 秒。
- 記憶體限制：32 MiB (1 MiB 為 1024x1024 bytes)
- $1 \leq H \leq 5{,}000$

計分方式

子任務	分數	額外輸入限制
1	12	已知開關 i 連到洞門 i。你只需要決定正確的開關設定位置。
2	13	正確的開關設定位置已知是 [0, 0, 0, …, 0]。你只需決定開關與洞門的連結關係。
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(無)*

實際演練

你電腦上的範例評分程式將會從 cave.in 讀入資料，其格式如下：

- 第一行：N
- 第二行：$S[0]$ $S[1]$ … $S[N-1]$
- 第三行：$D[0]$ $D[1]$ … $D[N-1]$

N 代表洞門和開關的數量，$S[i]$ 是開關 i 的正確設定位置，$D[i]$ 是開關 i 連結到的洞門編號。舉例來說，上面的範例會以下面的格式來提供資料：

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

程式語言附註

C/C++ 你必須 `#include "cave.h"`

Pascal 你必須定義 `unit Cave`，而且你也必須透過 `uses GraderHelpLib` 匯入 (import) 評分程序。所有的陣列元素從 0 開始編號。

請參照你的電腦上的解答樣板作為範例。