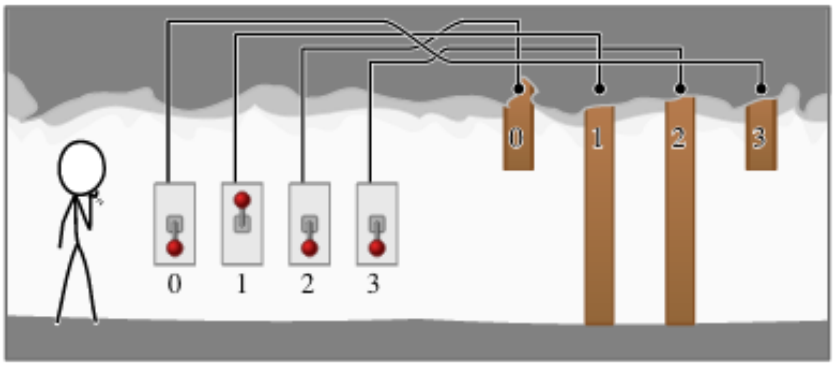
Ketika sedang tersesat dalam perjalanan panjang dari College ke UQ Center, Anda telah menemukan pintu masuk ke sebuah sistem gua rahasia yang berada di bawah universitas. Pintu masuk dihalangi oleh sebuah sistem pengamanan yang terdiri dari `N` buah pintu berurutan, satu pintu di belakang pintu yang lain, dan `N` buah saklar, dimana setiap saklar terhubung ke sebuah pintu.



Pintu-pintu diberi nomor `0, 1, …, (N - 1)` secara berurutan, dengan pintu 0 berada paling dekat dengan Anda. Saklar-saklar juga diberi nomor `0, 1, …, (N - 1)`, meskipun Anda tidak tahu saklar yang mana terhubung ke pintu yang mana.

Semua saklar berada pada pintu masuk gua. Setiap saklar dapat berada pada posisi "naik" atau "turun". Hanya salah satu dari posisi-posisi ini ("naik" atau "turun") yang sesuai untuk setiap saklar. Jika saklar berada pada posisi yang sesuai, maka pintu yang terhubung ke saklar itu akan terbuka, dan jika saklar berada pada posisi yang tidak sesuai, maka pintu yang terhubung ke saklar itu akan tertutup. Posisi yang sesuai berbeda-beda untuk setiap saklar, dan Anda tidak tahu posisi yang mana yang sesuai untuk setiap saklar.

Anda ingin memahami sistem pengamanan ini. Untuk melakukan ini, Anda dapat mengatur saklar-saklar tersebut ke kombinasi apapun, dan kemudian berjalan ke dalam gua untuk melihat pintu pertama yang tertutup. Pintu-pintu tersebut tidak transparan: ketika Anda menemui pintu pertama yang tertutup, Anda tidak dapat melihat ke pintu-pintu di belakang pintu yang tertutup tersebut.

Anda punya waktu untuk mencoba `70000` kombinasi saklar, tapi tidak lebih. Tugas Anda adalah menentukan posisi yang sesuai untuk setiap saklar, dan juga pintu yang mana yang terhubung ke setiap saklar.

Implementasi

Anda harus men-submit sebuah file yang mengimplementasikan prosedur `exploreCave()`. Prosedur ini boleh memanggil fungsi grader `tryCombination()` sampai maksimal 70000 kali, dan harus berakhir dengan memanggil prosedur grader `answer()`. Fungsi-fungsi dan prosedur-prosedur ini dijelaskan di bawah ini.

Fungsi grader: `tryCombination()`

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

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

Deskripsi

Grader akan menyediakan fungsi ini. Fungsi ini memungkinkan Anda untuk mencoba sebuah kombinasi saklar-saklar, dan kemudian memasuki gua untuk menentukan pintu pertama yang tertutup. Jika semua pintu terbuka, fungsi akan mengembalikan nilai `-1`. Fungsi ini akan berjalan dalam waktu `O(N)`; atau dengan kata lain, running-time terburuknya adalah proporsional terhadap `N`.

Fungsi ini boleh dipanggil sebanyak maksimal `70000` kali.

Parameter

- `S`: Sebuah array sepanjang `N`, menunjukkan posisi dari setiap saklar. Elemen `S[i]` menunjukkan posisi saklar nomor `i`. Nilai `0` menunjukkan bahwa saklar dalam posisi "naik", dan nilai `1` menunjukkan saklar dalam posisi "turun".
- *Returns*: Nomor pintu pertama yang tertutup, atau `-1` jika semua pintu terbuka.

Prosedur grader: `answer()`

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

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

Deskripsi

Panggil prosedur ini ketika Anda telah mengidentifikasi kombinasi semua saklar yang sesuai untuk membuka semua pintu, dan pintu-pintu mana yang terhubung ke setiap saklar.

Parameter

- `S`: Array sepanjang `N`, menyatakan posisi yang sesuai untuk setiap saklar. Formatnya sesuai dengan yang ada pada fungsi `tryCombination()` yang dideskripsikan di atas.
- `D`: Array sepanjang `N`, menyatakan pintu yang terhubung ke setiap saklar. Lebih spesifiknya, elemen `D[i]` harus mengandung nomor pintu yang terhubung dengan saklar nomor `i`.
- *Returns*: Prosedur ini tidak mengembalikan nilai, tetapi akan menyebabkan program exit.

Prosedur Anda: `exploreCave()`

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

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

Deskripsi

Submisi Anda harus mengimplementasikan prosedur ini.

Fungsi ini harus menggunakan rutin grader `tryCombination()` untuk menentukan posisi yamg benar untuk setiap saklar dan pintu yang terhubung ke setiap saklar, dan harus memanggil `answer()` ketika ia telah berhasil menentukan informasi ini.

Parameter

- `N`: Jumlah saklar dan pintu di dalam gua.

Contoh sesi

Misalkan pintu-pintu dan saklar-saklar diatur dalam posisi seperti gambar di atas.

Pemanggilan fungsi	Returns	Penjelasan
`tryCombination([1, 0, 1, 1])`	`1`	Ini sesuai dengan gambar di atas. Saklar 0, 2 dan 3 dalam posisi "turun", sementara saklar 1 dalam posisi "naik". Fungsi ini mengembalikan nilai `1`, menunjukkan bahwa pintu nomor 1 adalah pintu pertama dari kiri yang tertutup.
`tryCombination([0, 1, 1, 0])`	`3`	Pintu 0, 1 dan 2 terbuka semua, sedangkan pintu nomor 3 tertutup.
`tryCombination([1, 1, 1, 0])`	`-1`	Menggerakkan saklar nomor 0 "turun" ternyata menyebabkan semua pintu terbuka, ditunjukkan dengan nilai return `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program exit)*	Kita menebak bahwa kombinasi yang sesuai adalah `[1, 1, 1, 0]`, dan bahwa masing-masing saklar 0, 1, 2 dan 3 terhubung ke pintu 3, 1, 0 dan 2.

Batasan

- Time limit: 2 detik
- Memory limit: 32 MiB
- $1 \le N \le 5,000$

Subtasks

Subtask	Poin	Batasan Input Tambahan
1	12	Untuk setiap `i`, saklar `i` terhubung ke pintu `i`. Tugas Anda hanya menentukan kombinasi yang sesuai.
2	13	Kombinasi yang sesuai selalu `[0, 0, 0, …, 0]`. Tugas Anda hanya menentukan saklar mana terhubung ke pintu yang mana.
3	21	$N \le 100$
4	30	$N \le 2,000$
5	24	*(Tidak ada)*

Implementasi

Contoh grader pada komputer Anda akan membaca input dari file `cave.in`, yang harus dalam format berikut:

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

Disini `N` adalah jumlah saklar dan pintu, `S[i]` adalah posisi yang sesuai untuk saklar `i`, dan `D[i]` adalah pintu yang terhubung ke saklar `i`.

Misalnya, contoh di atas akan diberikan dalam format sebagai berikut:

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

Catatan Bahasa

C/C++	Anda harus `#include "cave.h"`.
Pascal	Anda harus mendefinisikan `unit Cave`, dan Anda juga harus import rutin grader dengan `uses GraderHelpLib`. Semua array dinomori mulai dari `0` (bukan `1`).

Lihat template solusi pada komputer Anda sebagai contoh.