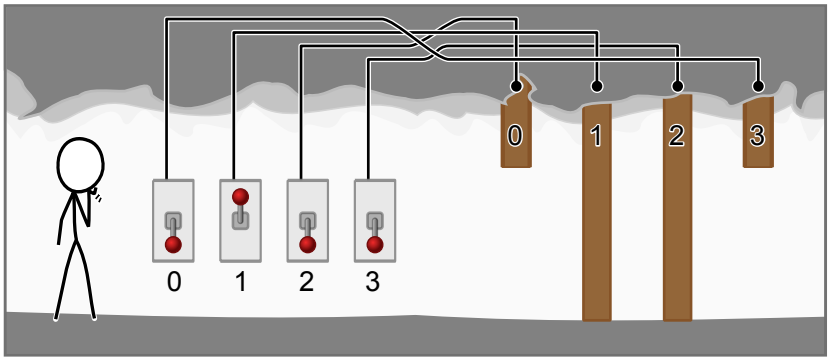
International Olympiad in Informatics 2013

July 2013 6-13
Brisbane, Australia
Day 2 tasks

غار

Persian — ۱.۰

بعد از گم کردن راه در یک پیاده‌روی طولانی از مرکز دانشگاه UQ، شما به ورودی یک مجموعه غار سحرآمیز رسیده‌اید که به اعماق زیر دانشگاه راه دارد. ورودی غار با یک سامانه‌ی امنیتی متشکل از N درب متوالی و N سوئیچ مسدود شده است. هر درب پشت درب قبلی قرار گرفته و هر سوئیچ به یک درب جداگانه وصل شده است (هیچ دو سوئیچی به یک درب متصل نیستند).



درب‌ها به ترتیب با اعداد 0, 1, ..., (N - 1) شماره‌گذاری شده‌اند، طوری که درب شماره‌ی 0 نزدیک‌ترین درب به شما می‌باشد. سوئیچ‌ها نیز با اعداد 0, 1, ..., (N - 1) شماره‌گذاری شده‌اند، اما شما نمی‌دانید که کدام سوئیچ به کدام درب متصل است.

سوئیچ‌ها در ورودی غار قرار گرفته‌اند. هر سوئیچ می‌تواند در یکی از دو حالت بالا یا پایین قرار داشته باشند. برای هر سوئیچ، تنها یکی از این حالات درست است: اگر یک سوئیچ در حالت درست خود باشد، دربی که به سوئیچ وصل است باز می‌شود و اگر سوئیچ در حالت درست خود نباشد، دربی که به آن وصل است، بسته خواهد بود. حالت درست می‌تواند برای سوئیچ‌های مختلف، متفاوت باشد و شما از حالت درست برای سوئیچ‌ها اطلاع ندارید.

شما می‌خواهید این سیستم امنیتی را بشناسید. برای این منظور، شما می‌توانید سوئیچ‌ها را در یک وضعیت دل‌خواه قرار دهید و سپس وارد غار شوید تا ببینید اولین درب بسته کدام است. درب‌ها شفاف نیستند و وقتی شما به اولین درب بسته برسید، هیچ دربی که پشت آن باشد را نمی‌بینید.

شما به اندازه‌ای زمان دارید که می‌توانید حداکثر 70,000 ترکیب از سوئیچ‌ها را بررسی کنید (و نه بیشتر). وظیفه شما این است که وضعیت درست هر سوئیچ و همچنین دربی که به آن سوئیچ متصل است را بیابید.

پیاده‌سازی

شما باید یک فایل شامل پیاده‌سازی تابع `exploreCave()` را به سامانه‌ی داوری ارسال کنید. این تابع می‌تواند حداکثر `70000` بار تابع `tryCombination()` را فراخوانی کند، و باید با فراخوانی تابع `answer()` در سامانه داوری به اتمام برسد. این توابع در ادامه توضیح داده شده‌اند.

تابع مصحح: `tryCombination()`

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

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

توضیحات

این تابع در برنامه‌ی مصحح پیاده‌سازی شده است. این تابع به شما اجازه می‌دهد که یک ترکیب از سوئیچ‌ها را امتحان کنید، و سپس با ورود به غار اولین درب بسته را پیدا کنید. اگر همه‌ی درب‌ها باز باشند، تابع مقدار `-1` را باز می‌گرداند. این تابع در زمان `O(N)` اجرا می‌شود، یعنی زمان اجرا در بدترین حالت متناسب با `N` خواهد بود.

این تابع باید حداکثر `70000` بار صدا زده شود.

پارامترها

- `S`: یک آرایه به طول `N` که وضعیت هر سوئیچ را مشخص می‌کند. `S[i]` مربوط به وضعیت سوئیچ `i`اُم است. مقدار `0` به معنی بالا بودن سوئیچ و مقدار `1` به معنی پایین بودن آن است.
- خروجی: شماره اولین دربی که بسته است. در صورت باز بودن همه درب‌ها مقدار `-1` برگردانده خواهد شد.

تابع مصحح: `answer()`

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

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

توضیحات

شما باید هنگامی این تابع را فراخوانی کنید که ترکیبی از سوئیچ‌ها که همه‌ی درب‌ها را باز می‌کند و همچنین شماره‌ی دربی که به هر سوئیچ متصل است را یافته باشید.

پارامترها

- `S`: یک آرایه به طول `N` که وضعیت درست هر سوئیچ را مشخص می‌کند. فرمت آن مشابه آن‌چه در تابع `tryCombination()` که در بالا توضیح داده شد می‌باشد.
- `D`: یک آرایه به طول `N` که درب متصل به هر سوئیچ را مشخص می‌کند. به طور مشخص، عنصر `D[i]` باید شامل شماره‌ی دربی باشد که سوئیچ `i`ام به آن متصل است.
- خروجی: این تابع چیزی باز نمی‌گرداند، اما باعث خاتمه‌ی برنامه می‌شود.

تابع شما: `exploreCave()`

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

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

توضیحات

برنامه‌ی ارسالی شما باید این تابع را پیاده‌سازی کند.

این تابع باید از تابع `tryCombination()` در برنامه‌ی مصحح، برای پیدا کردن وضعیت درست برای هر سوئیچ و درب متصل به آن، استفاده کند. بعد از تعیین این موارد، تابع `answer()` باید فقط یک بار فراخوانی شود.

پارامترها

- `N`: تعداد سوئیچ‌ها و درب‌های درون غار.

اجرای نمونه

فرض کنید درب‌ها و سوئیچ‌ها در وضعیتی که در شکل بالا نشان داده شده، قرار داشته باشند.

فراخوانی تابع	خروجی	توضیح
`tryCombination([1, 0, 1, 1])`	`1`	به شکل توجه کنید. سوئیچ‌های ۰، ۲ و ۳ پایین هستند، اما سوئیچ ۱ بالا است. تابع عدد ۱ را برمی‌گرداند، به این معنی که درب شماره‌ی ۱ اولین درب بسته از سمت چپ است.
`tryCombination([0, 1, 1, 0])`	`3`	درب‌های ۰، ۱ و ۲ باز هستند، درحالی که درب ۳ بسته است.
`tryCombination([1, 1, 1, 0])`	`-1`	پایین بردن سوئیچ ۰ باعث می‌شود که همه درب‌ها باز شوند و در نتیجه تابع عدد ۱- را برمی‌گرداند.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	(برنامه پایان می‌یابد)	حدس می‌زنیم که ترکیب درست سوئیچ‌ها `[1, 1, 1, 0]` است و سوئیچ‌های ۰، ۱، ۲ و ۳ به ترتیب به درب‌های ۳، ۱، ۰ و ۲ متصل هستند.

محدودیت‌ها

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۳۲ مگابایت
- $1 \le N \le 5,000$

زیرمسئله‌ها

زیرمسئله‌ها	امتیاز	محدودیت‌های اضافی ورودی
۱	۱۲	برای هر i، سوییچ i به درب i متصل است. وظیفه‌ی شما تنها پیدا کردن ترکیب درست برای سوئیچ‌ها است.
۲	۱۳	ترکیب درست همواره [0, 0, 0, …, 0] خواهد بود. شما تنها باید مشخص کنید که هر سوئیچ به کدام درب وصل است.
۳	۲۱	$N \le 100$
۴	۳۰	$N \le 2,000$
۵	۲۴	(بدون محدودیت اضافی)

آزمایش

مصححی که روی کامپیوتر شما قرار دارد، ورودی را از فایل cave.in می‌خواند، که این فایل باید به شکل زیر باشد:

- خط ۱: N
- خط ۲: S[0] S[1] … S[N - 1]
- خط ۳: D[0] D[1] … D[N - 1]

اینجا N تعداد درب‌ها و سوئیچ‌ها، S[i] وضعیت درست برای سوئیچ i و D[i] دربی است که سوئیچ i به آن متصل است.

برای نمونه، مثال بالا باید به شکل زیر داده شود:

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

نکات زبان

C/C++ شما باید `#include "cave.h"` را به برنامه‌ی خود اضافه کنید.

Pascal شما باید `unit Cave` را تعریف کنید و همچنین، توابع برنامه‌ی مصحح را به شکل زیر به برنامه‌ی خود اضافه کنید: `uses GraderHelpLib`. تمامی آرایه‌ها با شروع از `0` (و نه `1`) شمارش می‌شوند.

برای دیدن مثال‌ها به راه حل‌های نمونه بر روی کامپیوتر خود مراجعه کنید.