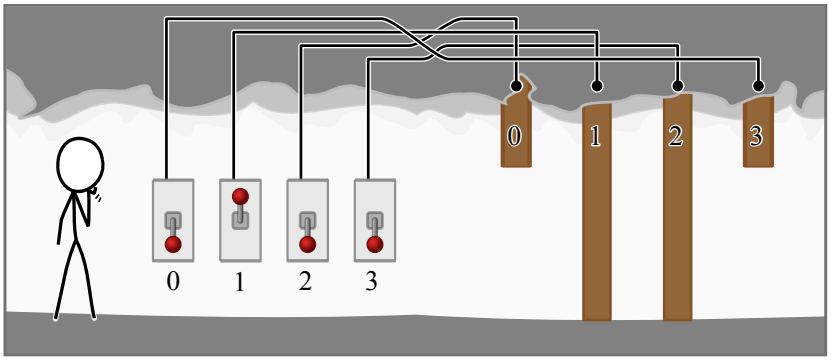
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Spanish — 1.0

Mientras estuvo perdido en la larga caminata desde el colegio al UQ Centre, usted se ha tropezado con la entrada secreta de un sistema de cuevas que funcionan muy abajo de la universidad. La entrada está bloqueada por un sistema de seguridad que consiste en `N` puertas consecutivas, cada puerta inmediatamente después de la anterior; y `N` interruptores, cada uno conectado a una puerta diferente.



Las puertas se encuentran numeradas `0, 1, …, (N - 1)` en orden, donde la puerta 0 es la más cercana a usted. Los interruptores también se encuentran numerados `0, 1, …, (N - 1)`, sin embargo, usted no sabe cual interruptor está asociado a cada puerta.

Todos los interruptores están ubicados en la entrada de la cueva. Cada interruptor puede estar en posición *arriba* o *abajo*. Sólo una de esas posiciones es correcta para cada interruptor. Si un interruptor está en la posición correcta entonces la puerta a la cual está conectado estará abierta, y si el interruptor está en la posición incorrecta la puerta a la cual está conectado estará cerrada. La posición correcta puede ser diferente para diferentes interruptores, y usted no sabe cuales posiciones son las correctas.

Usted quisiera entender este sistema de seguridad. Para lograrlo, usted puede configurar los interruptores a cualquier combinación, y luego caminar dentro de la cueva para observar cual es la primera puerta cerrada. Las puertas no son transparentes: una vez que usted encuentra la primera puerta cerrada, usted no puede ver el estado de las otras puertas detras de ésta.

Usted posee el tiempo para probar hasta `70,000` combinaciones para los interruptores, pero no más. Su tarea es determinar la posición correcta para cada interruptor, y además determinar a que puertas está conectado cada interruptor.

Implementación

Usted debe enviar un archivo que implemente el procedimiento `exploreCave()`. Este puede llamar a la función del evaluador `tryCombination()` hasta 70,000 veces, y debe finalizar invocando al procedimiento del evaluador `answer()`. Dichas funciones y procedimientos son descritas adelante.

Función del Evaluador: `tryCombination()`

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

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

Descripción

El evaluador proveerá dicha función. Ella permite probar una combinación de interruptores, y luego entrar a la cueva para determinar la primera puerta cerrada. Si todas las puertas están abiertas, la función va a retornar `-1`. Dicha función tiene complejidad en tiempo de `O(N)`; esto es, el tiempo de ejecución es proporcional al tamaño de `N`.

Esta función debe ser llamada a lo sumo `70,000` veces.

Parámetros

- `S`: Un arreglo de tamaño `N`, el cual indica la posición de cada interruptor. El elemento `S[i]` corresponde al interruptor `i`. Un valor de `0` indica que el interruptor está arriba, y un valor de `1` indica que el interruptor está abajo.
- *Retorna*: El número de la primera puerta que esté cerrada, o `-1` si todas las puertas están abiertas.

Procedimiento del Evaluador: `answer()`

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

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

Descripción

Llame a este procedimiento cuando halla identificado la combinación de interruptores que abren todas las puertas, y la puerta a la cual está conectado cada interruptor.

Parámetros

- `S`: Un arreglo de tamaño `N`, que indica la posición correcta para cada interruptor. El formato es el mismo de la función `tryCombination()` descrita anteriormente.
- `D`: Un arreglo de tamaño `N`, indicando la puerta a la cual está conectado cada interruptor. Especificamente, el elemento `D[i]` debe contener el número de puerta a la cual el interruptor `i` está conectado.
- *Retorno*: este procedimiento no retorna nada, pero permitirá finalizar la ejecución.

Su procedimiento: `exploreCave()`

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

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

Descripción

Su solución debe implementar dicho procedimiento.

Dicha función debe utilizar la rutina del evaluador `tryCombination()` para determinar la posición correcta de cada interruptor, la puerta a la cual está asociado cada interruptor, y debe llamar a `answer()` una vez que se halla determinado dicha información.

Parámetros

- `N`: El número de interruptores en la cueva.

Sesión de Ejemplo

Suponga que los dos interruptores están posicionados como se presenta en la figura siguiente:

LLamada a la función	Retorno	Explicación
`tryCombination([1, 0, 1, 1])`	`1`	Esto corresponde a la imagen. Los interruptores 0, 2 y 3 están abajo, mientras el interruptor 1 está arriba. La función retorna `1`, indicando que la puerta 1 es la primera puerta desde la izquierda que está cerrada.
`tryCombination([0, 1, 1, 0])`	`3`	Las puertas 0, 1 y 2 están todas abiertas, mientras la puerta 3 está cerrada.
`tryCombination([1, 1, 1, 0])`	`-1`	Mover el interruptor 0 hacia abajo causa que todas las puertas estén abiertas, indicado por el valor de retorno `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(El programa termina)*	Adivinamos que la combinación correcta es `[1, 1, 1, 0]`, y que los interruptores 0, 1, 2 y 3 se conectan a las puertas 3, 1, 0 y 2 respectivamente.

Restricciones

- Tiempo límite: 2 seconds
- Límite de memoria: 32 MiB
- $1 \le N \le 5{,}000$

Subtareas

Subtareas	Puntos	Restricciones adicionales de entrada
1	12	Por cada `i`, el switch `i` está conectado a la puerta `i`. Tu tarea es simplemente encontrar la combinación correcta.
2	13	La combinación correcta siempre será `[0, 0, 0, ..., 0]`. Tu tarea es simplemente determinar que switch está conectado a que puerta.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(None)*

Experimentación

El evaluador de ejemplo en tu computadora leerá la entrada desde el archivo `cave.in`, que debe estar en el siguiente formato.

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

Aquí `N` es el número de puertas e interruptores, `S[i]` es la posición correcta para el interruptor `i`, y `D[i]` es la puerta conectada al interruptor `i`

El ejemplo anterior sería provisto en el siguiente formato:

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

Notas del lenguaje

C/C++	Tu debes `#include "cave.h"`.
Pascal	Tu debes definir `unit Cave`, y además debe importar las rutinas del evaluador vía `uses GraderHelpLib`. Todos los arreglos son numerados empezando en `0` (no `1`).

Ve las plantillas de solución en tu máquina para ejemplos.