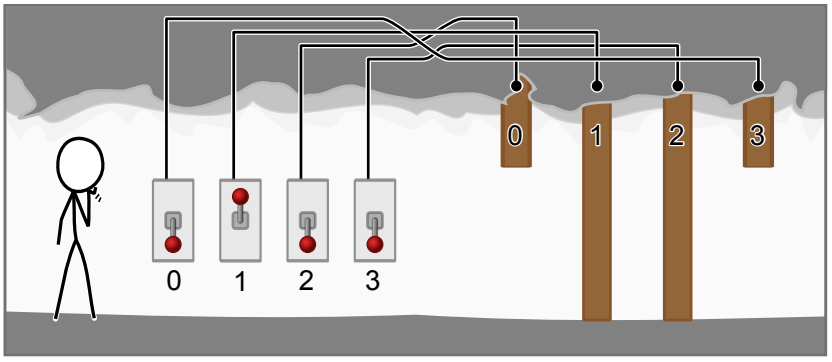
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Spanish — 1.0

Te has desviado accidentalmente del camino que hay entre la facultad y el UQ Centre, y has dado con la entrada a una cueva secreta que alguien construyó bajo el campus universitario. La entrada está bloqueada por un sistema de seguridad formado por `N` puertas consecutivas, cada una a continuación de la anterior, y `N` interruptores, cada uno de ellos conectado a una puerta diferente.



Las puertas están numeradas como `0, 1, ..., (N - 1)` en orden, donde la puerta 0 es la que tienes más cerca. Los interruptores también están numerados como `0, 1, ..., (N - 1)`, pero no sabes a qué puerta está conectado cada interruptor.

Todos los interruptores se encuentran en la entrada de la cueva. Cada interruptor puede estar en la posición de *arriba* o *abajo*. Por cada uno de los interruptores hay una única posición correcta. Si un interruptor está en su posición correcta entonces la puerta a la que está conectado estará abierta, y si el interruptor está en su posición incorrecta entonces la puerta a la que está conectado estará cerrada. Las posiciones correctas pueden ser diferentes para diferentes puertas, e inicialmente tu no sabes cuales son esas posiciones correctas.

Tienes curiosidad por entender el sistema de seguridad de la cueva. Para ello puedes cambiar la posición de los interruptores, dejándolos en cualquier combinación posible, y a continuación entrar en la cueva para ver cual es la primera puerta que se encuentra cerrada. Las puertas no son transparentes: una vez que encuentres la primera puerta cerrada no puedes ver ninguna de las puertas que hay detrás de esta.

Tienes tiempo para probar `70.000` combinaciones de interruptores, pero no más. Tu misión es determinar la posición correcta de cada interruptor, y también a qué puerta está conectado.

Implementación

Se te pide que envíes un archivo que implemente el procedimiento `exploreCave()`. Este procedimiento puede llamar a la función `tryCombination()` hasta un máximo de 70.000 veces, y debe acabar llamando al procedimiento `answer()`. A continuación se describen estas funciones y procedimientos.

Función del corrector: `tryCombination()`

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

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

Descripción

Esta función la proporciona el corrector. Te permite probar una combinación de interruptores, y a continuación entrar en la cueva para comprobar cual es la primera puerta que se encuentra cerrada. Si todas las puertas están abiertas la función devolverá `-1`. La función tarda `O(N)`; esto es, el tiempo de ejecución en el peor de los casos es proporcional a `N`.

Puedes llamar a esta función un máximo de `70.000` veces.

Parámetros

- `S`: Un array de longitud `N`, que indica la posición de cada interruptor. El elemento `S[i]` corresponde al interruptor `i`. El valor `0` indica que el interruptor está *arriba*, y el valor `1` indica que el interruptor está *abajo*.
- *Returns*: El número de la primera puerta que se encuentra cerrada, o bien `-1` si todas las puertas se encuentran abiertas.

Procedimiento del corrector: `answer()`

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

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

Descripción

Deberás llamar a este procedimiento cuando hayas identificado la combinación de interruptores que abre todas las puertas, y la puerta a la que está conectado cada interruptor.

Parámetros

- `S`: Un array de longitud `N` que indica la posición correcta de cada interruptor. El formato coincide con el de la función `tryCombination()` descrita anteriormente.
- `D`: Un array de longitud `N` que indica a qué puerta está conectado cada interruptor. Concretamente, el elemento `D[i]` debe contener el número de puerta a la que está conectado el interruptor `i`.
- *Returns*: Este procedimiento no devuelve nada, pero hará que el programa acabe.

Tu procedimiento: `exploreCave()`

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

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

Descripción

Tu envío debe implementar este procedimiento.

La función deberá usar la rutina del corrector `tryCombination()` para determinar la posición correcta de cada interruptor y la puerta a la que está conectado cada interruptor, y debe llamar a `answer()` una vez que haya determinado esta información.

Parámetros

- `N`: El número de interruptores y puertas de la cueva.

Secuencia de ejemplo

Supón que las puertas y los interruptores están dispuestos como en el dibujo que aparece en la descripción:

Llamada a función	Returns	Explicación
`tryCombination([1, 0, 1, 1])`	`1`	Esta combinación corresponde con la del dibujo. Los interruptores 0, 2 y 3 están abajo, mientras que el interruptor 1 está arriba. La función devuelve `1`, indicando que la puerta 1 es la primera puerta cerrada empezando por la izquierda.
`tryCombination([0, 1, 1, 0])`	`3`	Las puertas 0, 1 y 2 están ahora abiertas, mientras que la 3 está cerrada.
`tryCombination([1, 1, 1, 0])`	`-1`	Al bajar el interruptor 0 todas las puertas se abren, lo que se indica por el valor de retorno `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Programa finaliza)*	Deducimos que la combinación correcta es `[1, 1, 1, 0]`, y que los interruptores 0, 1, 2 y 3 están conectados con las puertas 3, 1, 0 y 2 respectivamente.

Restricciones

- Tiempo limite: 2 segundos
- Limite de memoria: 32 MiB
- $1 \le N \le 5{,}000$

Subtareas

Subtarea	Puntos	Restricciones adicionales de la entrada
1	12	Por cada `i`, el interruptor `i` está conectado a la puerta `i`. Tu tarea consiste únicamente en determinar cual es la combinación correcta.
2	13	La combinación correta siempre será `[0, 0, 0, …, 0]`. Tu tarea consiste únicamente en determinar a qué puerta está conectado cada interruptor.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(Ninguna)*

Implementación

El corrector de ejemplo que tienes disponible en tu ordenador lee la entrada desde el archivo `cave.in`, que debe estar en el formato siguiente:

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

`N` es el número de puertas e interruptores, `S[i]` es la posición correcta del interruptor `i`, y `D[i]` es la puerta a la que está conectado el interruptor `i`.

Por ejemplo, el ejemplo anterior se daría en el formato siguiente:

4

1 1 1 0

3 1 0 2

Apuntes del Lenguaje

Échale un vistazo a los moldes que hay en tu ordenador para ver ejemplos.