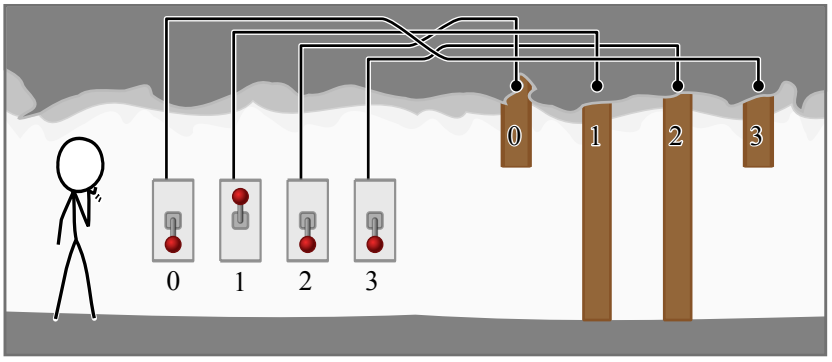
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cueva

Español-Colombia
— 1.0

En el largo camino entre su residenccia y el Centro de UQ, usted ha encontrado la entrada de una cueva secreta que alguien construyó bajo el campus universitario. La entrada está bloqueada por un sistema de seguridad formado por `N` compuertas consecutivas, cada una a continuación de la siguiente y `N` interruptores, cada uno de ellos conectado a una compuerta diferente.



Las compuertas están numeradas `0, 1, ..., (N - 1)`, donde la compuerta 0 es la que está más cerca a usted. Los interruptores también están numerados `0, 1, ..., (N - 1)`, pero usted no sabe a qué compuerta está conectado cada interruptor.

Todos los interruptores se encuentran en la entrada de la cueva. La posición de cada interruptor puede ser `arriba` o `abajo`. Para cada uno de los interruptores hay una única posición correcta. Si un interruptor está en la posición correcta entonces la compuerta a la que está conectado estará abierta, mientras que si el interruptor está en la posición incorrecta entonces la compuerta a la que está conectado estará cerrada. Las posiciones correctas pueden ser diferentes para interruptores diferentes y usted no sabe cuáles son las posiciones correctas.

Usted quiere entender este sistema de seguridad. Para hacer esto, usted puede cambiar la posición de los interruptores a cualquier combinación posible y a continuación entrar en la cueva para ver cuál es la primera compuerta que se encuentra cerrada. Las compuertas no son transparentes: una vez que encuentre la primera compuerta cerrada no podrá ver ninguna de las compuertas que hay detrás de esta.

Sólo tiene tiempo para probar `70.000` combinaciones de interruptores. Su tarea es determinar la posición correcta de cada interruptor y a qué compuerta está conectado.

Implementación

Usted debe enviar 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.

Calificador: `tryCombination()`

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

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

Descripción

El calificador le proporciona esta función. Le permite probar una combinación de interruptores y a continuación entrar en la cueva para comprobar cuál es la primera compuerta que se encuentra cerrada. Si todas las compuertas están abiertas la función retornará `-1`. El tiempo de ejecución de esta función es `O(N)`; es decir, el tiempo de ejecución es proporcional a `N` en el peor de los casos.

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

Parámetros

- `S`: Un arreglo 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 hacia arriba, y el valor `1` indica que el interruptor está hacia abajo.
- *Returns*: El número de la primera compuerta que se encuentra cerrada o bien `-1` si todas las puertas se encuentran abiertas.

Procedimiento del Calificador: `answer()`

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

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

Descripción

Debe llamar a este procedimiento cuando haya identificado la combinación de interruptores que abre todas las compuertas y la compuerta a la que está conectado cada interruptor.

Parámetros

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

Su procedimiento: `exploreCave()`

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

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

Descripción

Su envío debe implementar este procedimiento.

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

Parámetros

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

Sesión de Ejemplo

Suponga que las compuertas y los interruptores están dispuestos como en la figura mostrada anteriormente:

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

Restricciones

- Límite de tiempo: 2 segundos
- Límite de memoria: 32 MB
- $1 \le N \le 5{,}000$

Subtareas

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

Experimentación

El calificador en su máquina leerá de el archivo `cave.in`, que va 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]`

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

Así, el primer ejemplo iría en el siguiente formato:

4

1 1 1 0

3 1 0 2

Notas de Lenguaje

C/C++	Debe aparecer la instrucción `#include "cave.h"`.
Pascal	Debe definir `unit Cave`, y usted debe importar también las rutinas del calificador via `uses GraderHelpLib`. Todos los arreglos estan numerados y comienzan con `0` (no con `1`).

Para más información vea los esqueletos de solución en su máquina.