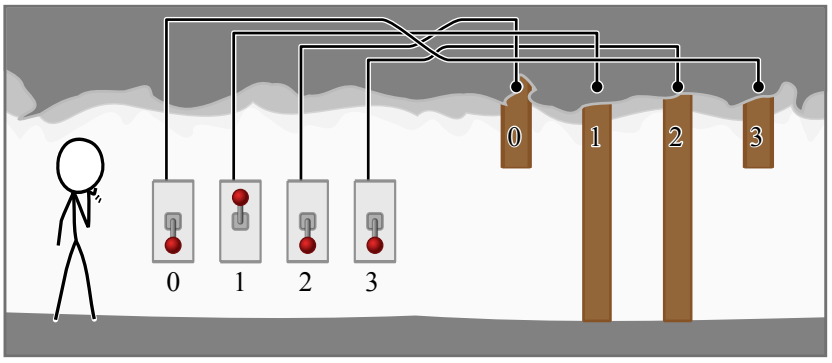
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Español-ARG — 1.0

Habiendote extraviado en la larga caminata desde la residencia (college) hasta el Centro UQ, has tropezado a través de la entrada a un sistema secreto de grutas que se extiende profundamente bajo la universidad. La entrada está obstruida por un sistema de seguridad consistiendo en `N` puertas consecutivas, cada puerta detrás de las previas; y `N` llaves, cada una de las cuales conectada a una puerta diferente.



Las puertas están numeradas `0, 1, …, (N - 1)` en orden, siendo la puerta 0 la primera con la que te encuentras. Las llaves también están numeradas `0, 1, …,(N - 1)`, pero no sabes que llave está conectada a que puerta.

Las llaves están todas ubicadas a la entrada de la cueva. Cada llave puede estar en una posición *up* o *down* . Sólo una de esas posiciones es la correcta para cada llave. Si una llave está en la posición correcta la puerta a la cual está conectada estará abierta, y si está en la posición incorrecta entonces la puerta a la cual está conectada estará cerrada. La posición correcta puede ser diferente para llaves diferentes, y no sabes que posiciones son las correctas.

Desearías entender este sistema de seguridad. Para lograrlo, puedes poner las llaves en cualquier combinación y después caminar en la caverna para ver cual es la primera puerta cerrada. Las puertas no son transparentes: una vez que te encuentras con la primera cerrada, no puedes ver ninguna de las que están detrás.

Dispones de suficiente tiempo para ensayar `70.000` combinaciones de las llaves, pero no más. Tu tarea consiste en determinar la posición correcta de cada llave y también establecer para cada llave a que puerta está conectada.

Implementación

Deberías enviar un archivo que implemente el procedimiento `exploreCave()`. Este puede llamar a la función `tryCombination()` del calificador hasta 70.000 veces, y debe finalizar llamando al procedimiento `answer()` del calificador. Estas funciones y procedimientos son descritas a continuación.

Función `tryCombination()` del calificador

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

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

Descripción

El calificador proveerá esta función. Ella te permite probar una combinación de llaves, y luego ingresar a la cueva para determinar cuál es la primera puerta cerrada. Si todas las puertas están abiertas, la función retornará `-1`. Esta función corre en tiempo `O(N)`; esto es, el tiempo de ejecución es en el peor caso proporcional a `N`.

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

Parámetros

- `S`: Un arreglo de largo `N`, indicando la posición de cada llave. El elemento `S[i]` corresponde a la llave `i`. Un valor de `0` indica que la llave está up, y un valor of `1` indica que la llave está down.
- *Returns*: El número de la primera puerta que está cerrada, o `-1` si todas las puertas están abiertas.

Procedimiento `answer()` del calificador

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

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

Descripción

Llama este procedimiento cuando hayas identificado la combinación de llaves que abre todas las puertas, y la puerta a la cual está cada llave conectada.

Parámetros

- `S`: Un arreglo de largo `N`, indicando la posición correcta de cada llave. El formato se corresponde con el de la función `tryCombination()` descripta más arriba.
- `D`: Un arreglo de largo `N`, indicando la puerta a la cual cada llave está connectada. Especificamente, el elemento `D[i]` contendrá el número de puerta a la cual que llave `i` está connectada.
- *Retorno*: Este procedimiento no retorna, pero causará que el program termine.

Tu Procedimiento: `exploreCave()`

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

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

Descripción

Tu envío debe implantar este procedimiento.

Esta función usará la rutina `tryCombination()` del calificador para determinar la correcta posición de cada llave y la puerta a la cual cada llave está connecteda, y debe llamar al procedimiento `answer()` una vez que has determinado esta información.

Parámetros

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

Sesión de muestra

Suponga las puertas y llaves están dispuestas como en la figura arriba:

Llamada a función	Returns	Explicación
`tryCombination([1, 0, 1, 1])`	`1`	Esta combinación se corresponde con la del dibujo. Las llaves 0, 2 y 3 están bajadas, mientras que la llave 1 está subida. 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 la llave 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)*	He deducido que la combinación correcta es `[1, 1, 1, 0]`, y que las llaves 0, 1, 2 y 3 están conectadas con las puertas 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`, la llave `i` está conectada 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 conectada cada llave.
3	21	$N \le 100$
4	30	$N \le 2.000$
5	24	*(Ninguna)*

Experimentación

El calificador provisto en tu máquina leerá la entrada del 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 y llaves, `S[i]` es la posición correcta de la llave `i`, y `D[i]` es la puerta a la que está conectada la llave `i`.

Así, el ejemplo anterior debería ser provisto en el siguiente formato:

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

Notas de Lenguaje

C/C++	Debes incluir la instrucción `#include "cave.h"`.
Pascal	Debes definir `unit Cave`, y también debes importar las rutinas del corrector via `uses GraderHelpLib`. Todos los arrays están numerados, y empiezan en `0` (no en `1`).

Mira las solution template en tu ordenador para encontrar ejemplos.