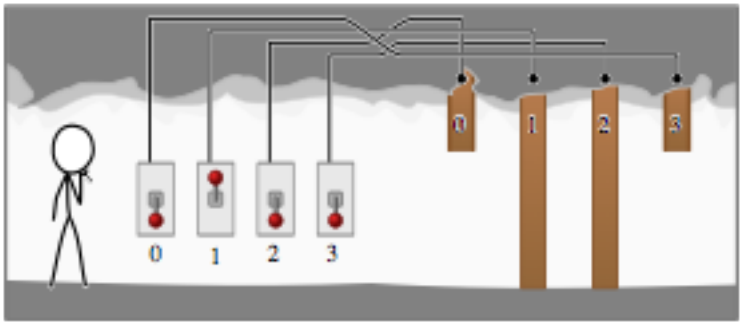
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Nederlands (België)
— 1.0

Je loopt verloren op de lange wandeling van het college naar het UQ Centre, en je stoot op de ingang van een geheim grottensysteem dat diep onder de universiteit loopt. De ingang is geblokkeerd door een veiligheidssysteem dat bestaat uit `N` opeenvolgende deuren, allemaal achter elkaar; en `N` schakelaars, waarbij elke schakelaar verbonden is met een verschillende deur.



De deuren zijn genummerd `0, 1, ..., (N - 1)` in volgorde, met deur 0 het dichtst bij jou. De schakelaars zijn ook genummerd `0, 1, ..., (N - 1)`, maar je weet niet welke schakelaar met welke deur verbonden is.

De schakelaars bevinden zich allemaal aan de ingang van de grot. Elke schakelaar kan zich in de stand *omhoog* of *omlaag* bevinden. Slechts één van deze standen is juist voor elke schakelaar. Als een schakelaar in de juiste stand staat, dan zal de deur waarmee hij verbonden is open zijn, en als de schakelaar in de foute stand staat dan zal de deur waarmee hij verbonden is gesloten zijn. De juiste stand kan verschillend zijn voor verschillende schakelaars, en je weet niet welke standen de juiste zijn.

Je wil dit veiligheidssysteem begrijpen. Om dat te doen, kan je de schakelaars in eender welke combinatie zetten, om daarna de grot in te wandelen om te zien welke de eerste gesloten deur is. Deuren zijn niet transparant: eens je op de eerste gesloten deur botst, kan je de deuren erachter niet zien.

Je hebt tijd om `70,000` combinaties van schakelaars uit te proberen, niet meer. Je taak is om te bepalen wat de correcte stand voor elke schakelaar is, en ook welke schakelaar met welke deur verbonden is.

Implementatie

Je moet een programma indienen dat de procedure `exploreCave()` implementeert. Deze procedure mag de volgende functie van de grader: `tryCombination()` maximaal 70,000 keer aanroepen. Als laatste moet je de volgende procedure van de grader aanroepen: `answer()`. Deze functies en procedures worden hieronder beschreven.

Grader Functie: `tryCombination()`

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

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

Beschrijving

Deze functie wordt door de grader voorzien. Ze laat je toe om een combinatie van schakelaars uit te proberen, om daarna de grot binnen te gaan en de eerste gesloten deur te bepalen. Als alle deuren open zijn dan geeft de functie `-1` als resultaat. Deze functie is van orde `O(N)`; d.w.z. de runtime is in het slechtste geval evenredig met `N`.

Je mag deze functie maximaal `70,000` keer aanroepen.

Parameters

- `S`: Een array van lengte `N`, waarin de stand van elke schakelaar staat aangegeven. Het element `S[i]` komt overeen met schakelaar `i`. De waarde `0` geeft aan dat de schakelaar omhoog staat. De waarde `1` geeft aan dat de schakelaar omlaag staat.
- *Resultaat*: Het nummer van de eerste deur die gesloten is, of `-1` als alle deuren openstaan.

Grader Procedure: `answer()`

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

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

Beschrijving

Roep deze procedure aan als je hebt ontdekt wat de juiste combinatie van schakelaars is om alle deuren te openen, en je ook weet bij welke deur elke schakelaar hoort.

Parameters

- `S`: Een array van lengte `N`, die aangeeft wat de juiste stand van elke schakelaar is. Het formaat is gelijk aan die in de functie `tryCombination()` zoals hierboven beschreven.
- `D`: Een array van lengte `N`, die aangeeft welke deur bij welke schakelaar hoort. Element `D[i]` moet het nummer van de deur bevatten waarmee schakelaar `i` is verbonden.
- *Resultaat*: Je krijgt geen resultaat. Deze procedure beëindigt jouw programma.

Jouw Procedure: `exploreCave()`

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

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

Beschrijving

Jouw inzending moet deze procedure implementeren.

Deze functie moet de grader routine `tryCombination()` gebruiken om te bepalen wat de juiste stand voor elke schakelaar is, en welke deur bij welke schakelaar hoort. Deze functie moet `answer()` aanroepen zodra je deze informatie hebt bepaald.

Parameters

- `N`: Het aantal schakelaars en deuren in de grot.

Voorbeeldscenario

Stel dat de deuren en schakelaars verbonden zijn zoals in de figuur hierboven.

Functie-aanroep	Resultaat	Uitleg
`tryCombination([1, 0, 1, 1])`	`1`	Dit komt overeen met de figuur. Schakelaars 0, 2 en 3 staan omlaag; schakelaar 1 staat omhoog. De functie geeft als resultaat `1` om aan te te geven dat deur 1 de eerste gesloten deur is die je tegenkomt.
`tryCombination([0, 1, 1, 0])`	`3`	De deuren 0, 1 en 2 zijn nu open; deur drie is gesloten.
`tryCombination([1, 1, 1, 0])`	`-1`	Door schakelaar 0 ook omlaag te zetten kan je alle deuren openzetten. Dit zie je aan het resultaat `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(programma eindigt)*	We denken dat de correcte combinatie `[1, 1, 1, 0]` is, en dat de schakelaars 0, 1, 2 en 3 verbonden zijn met respectievelijk deuren 3, 1, 0 en 2.

Beperkingen

- Tijdslimiet: 2 seconden
- Geheugenlimiet: 32 MiB
- $1 \le N \le 5{,}000$

Subtaken

Subtaak	Punten	Aanvullende beperkingen
1	12	Voor alle `i`, is schakelaar `i` verbonden met deur `i`. Je hoeft alleen te bepalen wat de juiste combinatie is.
2	13	De juiste combinatie is altijd `[0, 0, 0, …, 0]`. Je hoeft alleen te bepalen welke deur bij welke schakelaar hoort.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(geen)*

Experimenteren

De voorbeeldgrader op jouw computer leest invoer uit het bestand `cave.in`, dat in het volgende formaat is opgesteld:

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

`N` is het aantal deuren en schakelaars, `S[i]` is de juiste stand voor schakelaar `i`, en `D[i]` is de deur die hoort bij schakelaar `i`.

Het bovenstaande voorbeeld is dus in het volgende formaat opgesteld:

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

Taalspecifieke opmerkingen

C/C++	Je moet `#include "cave.h"` gebruiken.
Pascal	Je moet `unit Cave` definiëren, en de routines van de grader importeren door `uses GraderHelpLib` te gebruiken. Alle arrays zijn genummerd vanaf `0` (niet `1`).

Bekijk de modeloplossing op je computer als voorbeeld.