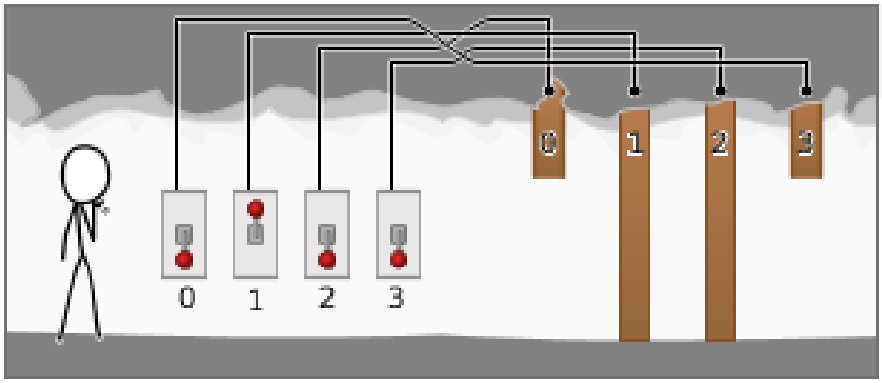
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave
Bulgarian — 1.0

Намирате се пред вход. Входът е блокиран чрез система от N последователно разположени врати. Има N ключа. Всеки ключ е свързан с различна врата. Вратите са номерирани 0, 1, ..., N-1. Вратата с номер 0 е най-близката до вас. Ключовете имат номера 0, 1, ..., N-1, но не се знае кой ключ за коя врата е.



Ключовете са разположени пред входа. Всеки ключ може да бъде само в две позиции: up или down. Само една от тези позиции е правилна. Ако ключът е в правилна позиция, тогава вратата, която е свързана към него, ще бъде отворена. Ако ключът е в неправилна позиция, тогава вратата, която е свързана към него, ще бъде затворена. Правилната позиция може да е различна за различните ключове. Вие не знаете кои позиции са правилни.

Вие искате да разгадаете цялата заключваща система. За да направите това, можете да превключвате ключовете на всички позиции, които изберете и да влизате и да видите, коя е първата затворена врата. Вратите не са прозрачни, т.е. не може да видите кои врати са отворени или затворени след първата затворена врата.

Имате възможност да пробвате най-много 70 000 комбинации в позициите на ключовете. Вашата задача е да намерите правилната позиция за всеки ключ и кой ключ към коя врата е свързан.

Имплементация

Трябва да изпратите файл, съдържащ сорса на процедурата `exploreCave()`. Тя може да извиква фунцията на грейдера `tryCombination()` най-много 70 000 пъти и трябва да

завърши с извикване на фунцията на грейдера `answer()`. Тези функции и процедури са описани по-долу.

Функция на грейдера: `tryCombination()`

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

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

Описание

Тази функция се осигурява от грейдера. Тя позволява да експериментирате комбинациите от ключове и да влизате във входа, за да видите, коя е първата затворена врата. Ако всичките врати са отворени, функцията връща `-1`. Тази функция работи за **O(N)** време, т.е. времето за работа в най-лошия случай е пропорционално на **N**.

Тази функция може да бъде извикана най-много **70 000** пъти.

Параметри

- `S`: масив с дължина **N**, показващ позицията на всеки ключ. Елементът **S[i]** съответства на ключ **i**. Стойност `0` показва, че ключът е up, а стойност `1` показва, че ключът е down.
- *Returns*: Номерът на първата затворена врата или `-1,` ако всички врати са отворени.

Процедура на грейдера: `answer()`

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

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

Описание

Трябва да извикате тази процедура, когато сте намерили комбинацията на ключовете, с която се отварят всички врати и сте намерили кой ключ за коя врата е.

Параметри

- `S`: Масив с дължина **N**, показващ правилните позиции на всеки ключ. Форматът е същият, както при функцията `tryCombination(),` по-горе..
- `D`: Масив с дължина **N**, показващ вратата, за която е ключът. По-точно, елементът **D[i]** трябва да съдържа номерът на вратата, към която ключ **i** е свързан.
- *Returns*: няма; тази процедура завършва вашата програма.

Ваша процедура: `exploreCave()`

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

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

Описание

Вие трябва да напишете тази процедура.

Тази функция трябва да съдържа функцията на грейдера `tryCombination()`, за да определи правилната позиция за всеки ключ и да определи кой ключ за коя врата е. Тази ваша функция трябва извика веднъж функцията на грейдера `answer()`, за да завърши.

Параметри

- `N`: Брой на ключовете и вратите.

Примерна сесия

Предполагаме, че вратите и ключовете са подредени и свързани, както е на картинката по-горе.

Function Call	Returns	Explanation
`tryCombination([1, 0, 1, 1])`	`1`	Това съответства на картинката. Ключове 0, 2 и 3 са down, а ключ 1 е up. Тази функция връща `1`, с което показва, че врата 1 е първата (от ляво) врата, която е затворена.
`tryCombination([0, 1, 1, 0])`	`3`	Врати 0, 1 и 2 са отворени, а врата 3 е затворена.
`tryCombination([1, 1, 1, 0])`	`-1`	Превключването на ключ 0 на down предизвиква всички врати да са отворени. Така функцията връща `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program exits)*	Досещаме се, че правилната комбинация е `[1, 1, 1, 0]` и че ключове 0, 1, 2 и 3 са свързани, съответно с врати 3, 1, 0 и 2.

Ограничения

- Time limit: 2 seconds
- Memory limit: 32 MiB
- **`1 ≤ N ≤ 5,000`**

Подзадачи

Subtask	Points	Допълнителни ограничения за входа
1	12	За всяко **`i`**, ключ **`i`** е свързан с врата **`i`**. Вашата задача е просто да определите правилната комбинация.
2	13	Правилната комбинация винаги е `[0, 0, 0, …, 0]`. Вашата задача е просто да определите кой ключ към коя врата е свързан.

3	21	**N ≤ 100**
4	30	**N ≤ 2,000**
5	24	*Няма допълнителни ограничения*

Експериментиране

Опростеният грейдер на вашия компютър ще чете входа от файл `.in`, който трябва да е написан в следния формат:

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

Тук `N` е броят на вратите и ключовете, `S[i]` е правилната позиция на ключ **i** и `D[i]` е вратата към която е свързан ключ **i**.
За примера по-горе файлът съдържа:

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

Бележки за езиците за програмиране

C/C++ Трябва да използвате `#include "cave.h"`.

Pascal Трябва да дефинирате `unit Cave` и трябва да направите import на функциите на грейдера чрез `uses GraderHelpLib`. Всички масиви започват с индекс `0`.

Вижте за пример темплейтите на вашия компютър.