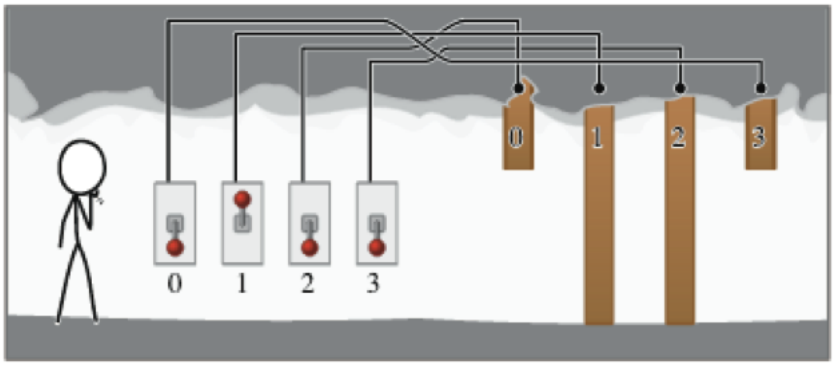
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Slovenščina — 1.0

Na poti proti UQ centru si se izgubil in po naključju zataval k vhodu v skrivni podzemni sistem jam, ki se nahaja globoko pod univerzo. Vhod varuje varnostni sistem `N` zaporednih vrat (ena za drugimi) in `N` stikal. Vsako stikalo je povezano z enimi izmed vrat.



Vrata so po vrsti oštevilčena `0, 1, …, (N - 1)`. Stojiš pred vrati s številko 0. Stikala so ravno tako oštevilčena `0, 1, …, (N - 1)`, žal pa ne veš, katero stikalo je povezano s katerimi vrati.

Vsa stikala se nahajajo pri vhodu v jamo. Vsako stikalo je lahko v položaju *gor* ali *dol*. Samo eden izmed teh položajev je pravilen za posamezna vrata. Če je stikalo v pravem položaju, bodo pripadajoča vrata odprta, v nasprotnem primeru pa zaprta. Pravilni položaj je za vsaka vrata lahko drugačen. Pravilnih položajev ne poznaš.

Rad bi razumel pričujoči varnostni sistem. Da bi to dosegel, lahko stikala nastaviš na poljubno kombinacijo, nato pa zakoračiš v jamo in pogledaš, katera vrata so prva zaprta. Vrata niso prosojna: ko se znajdeš pred prvimi zaprtimi vrati, skoznje ne vidiš ničesar.

Preizkusiš lahko največ `70 000` kombinacij stikal. Tvoja naloga je določiti pravilne položaje vseh stikal ter povezave med stikali in vrati (katera vrata pripadajo posameznim stikalom).

Implementacija

Oddati moraš datoteko, v kateri je implementirana procedura `exploreCave()`. Procedura lahko do `70 000`-krat kliče ocenjevalnikovo funkcijo `tryCombination()`. Zaključiti se mora s klicem ocenjevalnikove procedure `answer()`. Funkcija `tryCombination()` in procedura `answer()` sta opisani v nadaljevanju.

Ocenjevalnikova funkcija: `tryCombination()`

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

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

Opis

To funkcijo nudi ocenjevalnik. Omogoča preizkus posamezne kombinacije stikal in vstop v jamo, da odkriješ prva zaprta vrata. Če so vsa vrata odprta, funkcija vrne `-1`. Funkcija zahteva čas reda `O(N)` ; v najslabšem primeru je izvajalni čas premosorazmeren `N`.

Funkcijo se lahko kliče največ `70 000`-krat.

Parametri

- `S`: Polje velikosti `N`, ki določa položaje vseh stikal. Element `S[i]` ustreza stikalu `i`. Vrednost `0` predstavlja položaj *gor*, vrednost `1` pa položaj *dol*.
- *Vrača*: Številko prvih zaprtih (vidnih) vrat oziroma `-1`, če so vsa vrata odprta.

Ocenjevalnikova procedura: `answer()`

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

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

Opis

Proceduro pokliči šele takrat, ko poznaš pravilno kombinacijo stikal in pripadnosti vrat stikalom.

Parametri

- `S`: Polje velikosti `N`, ki predstavlja pravilne položaje vseh stikal. Oblika je enaka kot pri zgoraj opisani funkciji `tryCombination()`.
- `D`: Polje velikosti `N`, ki predstavlja pripadajoča vrata za vsako stikalo. Element `D[i]` vsebuje številko vrat, s katerimi je povezano stikalo `i`.
- *Vrača*: Procedura ne vrne ničesar, povzroči pa zaključitev programa.

Tvoja procedura: `exploreCave()`

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

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

Opis

Procedura naj s pomočjo ocenjevalnikove funkcije `tryCombination()` ugotovi pravilne položaje stikal in stikalom pripadajoča vrata. Ko ugotovi iskane podatke, mora poklicati proceduro `answer()`, nakar se izvajanje programa zaključi.

Parametri

- `N`: Število stikal in vrat v jami.

Vzorčno zaporedje klicev

Predpostavi, da so vrata in stikala razporejena tako, kot prikazuje zgornja slika:

Klic	Vrača	Razlaga
`tryCombination([1, 0, 1, 1])`	`1`	Ta klic ustreza sliki. Stikala 0, 2 and 3 so v položaju *dol*, stikalo 1 pa v položaju *gor*. Funkcija vrne `1`, kar pomeni, da so prva zaprta vrata tista s številko 1.
`tryCombination([0, 1, 1, 0])`	`3`	Vrata 0, 1 in 2 so odprta, vrata 3 pa zaprta.
`tryCombination([1, 1, 1, 0])`	`-1`	Po premiku stikala 0 navzdol so vsa vrata odprta, kar predstavlja vrnjena vrednost `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program se zaključi)*	Ugotovili smo, da je pravilna kombinacija stikal `[1, 1, 1, 0]` in da stikalom 0, 1, 2, 3 po vrsti pripadajo vrata 3, 1, 0, 2.

Omejitve

- Časovna omejitev: 2 sekundi
- Prostorska omejitev: 32 MiB
- $1 \le N \le 5\,000$

Podnaloge

Podnaloga	Točke	Dodatne omejitve vhoda
1	12	Za vsak `i` je stikalo `i` povezano z vrati `i`. Tvoja naloga je zgolj ugotoviti pravilne položaje stikal.
2	13	Pravilni položaji bodo vedno `[0, 0, 0, …, 0]`. Tvoja naloga je zgolj ugotoviti pripadajoča vrata.
3	21	$N \le 100$
4	30	$N \le 2\,000$
5	24	*(Brez)*

Preizkušanje

Vzorčni ocenjevalnik na tvojem računalniku bere vhod iz datoteke `cave.in`, ki mora biti sledeče oblike:

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

Tu je `N` število vrat in stikal, `S[i]` je pravilni položaj stikala `i`, `D[i]` pa so vrata, povezana s stikalom `i`.

Zgornji primer bi bil podan v sledeči obliki:

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

Jezikovne opombe

C/C++	Potrebuješ `#include "cave.h"`.
Pascal	Definiraj `unit Cave`, in vključi ocenjevalnikove rutine z `uses GraderHelpLib`. Oštevilčenje vseh polj se prične z `0` in ne z `1`.

Za primere si oglej predloge rešitev na tvojem računalniku.