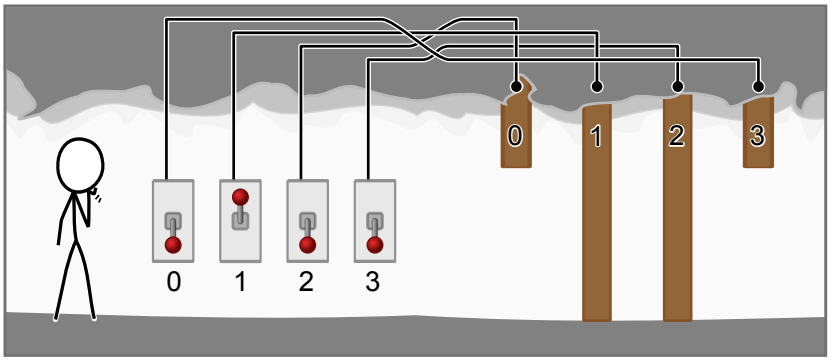
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Greek — 1.0

Καθώς ήσουν χαμένος σε ένα μακρινό περίπατο από ένα κολλέγιο προς το κέντρο του UQ, έπεσες πάνω στην είσοδο ενός συστήματος σπηλιών που εκτείνεται κάτω από το πανεπιστήμιο. Η είσοδος είναι μπλοκαρισμένη από ένα σύστημα ασφαλείας το οποίο αποτελείται από N διαδοχικές πόρτες οι οποίες βρίσκονται η κάθε μια πίσω από την προηγούμενη και N διακόπτες, ο καθένας από τους οποίους είναι συνδεδεμένος με μια διαφορετική πόρτα.



Οι πόρτες είναι αριθμημένες διαδοχικά 0, 1, …, (N - 1) με την πόρτα με αριθμό 0 να είναι η πλησιέστερη προς εσένα. Οι διακόπτες είναι επίσης αριθμημένοι 0, 1, …, (N - 1), αλλά δεν γνωρίζεις ποιός διακόπτης είναι συνδεδεμένος με ποιά πόρτα.

Όλοι οι διακόπτες είναι τοποθετημένοι στην είσοδο της σπηλιάς. Κάθε διακόπτης μπορεί να είναι στη θέση *up* ή *down*. Μόνο μια από τις θέσεις είναι η σωστή για τον κάθε διακόπτη. Αν ένας διακόπτης είναι στη σωστή θέση τότε η πόρτα με την οποία είναι συνδεδεμένος θα είναι ανοικτή, ενώ αν ο διακόπτης είναι στη λάθος θέση η πόρτα με την οποία είναι συνδεδεμένος θα είναι κλειστή. Η ορθή θέση μπορεί να είναι διαφορετική για τους διαφορετικούς διακόπτες και δεν γνωρίζεις ποιές θέσεις είναι οι ορθές.

Θέλεις να κατανοήσεις αυτό το σύστημα ασφαλείας. Για να το κάνεις αυτό, μπορείς να βάλεις τους διακόπτες με οποιοδήποτε συνδυασμό και μετά να προχωρήσεις μέσα στο σπήλαιο για να δείς ποιά είναι η πρώτη κλειστή πόρτα. Οι πόρτες δεν είναι διαφανείς, μόλις συναντήσεις την πρώτη κλειστή πόρτα δεν μπορείς να δεις τις πόρτες πίσω της.

Έχεις χρόνο για να δοκιμάσεις 70,000 συνδυασμούς από διακόπτες αλλά όχι περισσότερους. Ζητείται να προσδιορίσεις τη σωστή θέση κάθε διακόπτη και επίσης ποιά πόρτα είναι συνδεδεμένη με τον κάθε διακόπτη.

Υλοποίηση

Πρέπει να υποβαλεις ένα αρχείο που υλοποιεί τη διαδικασία `exploreCave()`. Αυτό μπορεί να καλέσει τη συνάρτηση βαθμολόγησης `tryCombination()` μέχρι και 70,000 φορές, και πρέπει να τελειώσει καλώντας τη διαδικασία βαθμολόγησης `answer()`. Αυτές οι διαδικασίες και συναρτήσεις περιγράφονται πιό κάτω.

Συνάρτηση βαθμολόγησης (grader): `tryCombination()`

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

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

Περιγραφή

Ο βαθμολογητής (grader)θα κάνει την εξής λειτουργία. Σου επιτρέπει να δοκιμάσεις ένα συνδυασμό από διακόπτες και μετά να μπεις στη σπηλιά για να ενοπίσεις την πρώτη κλειστή πόρτα. Αν όλες οι πόρτες είναι ανοικτές η συνάρτηση θα επιστρέψει `-1`. Αυτή η συνάρτηση τρέχει σε χρόνο `O(N)`, δηλαδή ο χρόνος εκτέλεσης είναι στη χειρότερη περίπτωση ανάλογος του `N`.

Αυτή η συνάρτηση μπορεί να κληθεί το πολύ `70,000` φορές.

Παράμετροι

- `S`: Πίνακας μεγέθους `N`, που δείχνει τη θέση του κάθε διακόπτη. Το στοιχείο `S[i]` αντιστοιχεί στο διακόπτη `i`. Η τιμή `0` σημαίνει ότι ο διακόπτης είναι πάνω (up), ενώ η τιμή `1` σημαίνει ότι ο διακόπτης είναι κάτω (down).
- *Returns*: Τον αριθμό της πρώτης πόρτας που είναι κλειστή ή `-1` αν όλες οι πόρτες είναι ανοικτές.

Διαδικασία Grader: `answer()`

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

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

Περιγραφή

Κάλεσε αυτή τη διαδικασία όταν έχεις προσδιορίσει το συνδυασμό των διακοπτών με τον οποίο ανοίγουν όλες οι πόρτες και τη πόρτα με την οποία είναι συνδεδεμένος ο κάθε διακόπτης.

Παράμετροι

- `S`: Πίνακας μεγέθους `N`, που δείχνει τη σωστή θέση για κάθε διακόπτη. Η μορφοποίηση πρέπει να είναι η ίδια με αυτή της συνάρτησης `tryCombination()` που περιγράφεται πιο πάνω.
- `D`: Πίνακας μεγέθους `N`, που δείχνει την πόρτα με την οποία είναι συνδεδεμένος ο κάθε διακόπτης. Συγκεκριμένα, το στοιχείο `D[i]` πρέπει να περιέχει τον αριθμό της πόρτας με την οποία είναι συνδεδεμένος ο διακόπτης `i`.
- *Returns*: Η διαδικασία αυτή δεν επιστρέφει κάτι, απλώς οδηγεί το πρόγραμμα σε έξοδο.

Η διαδικασία: `exploreCave()`

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

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

Περιγραφή

Η υποβολή σας πρεπει να υλοποιεί αυτή τη διαδικασία.

Αυτή η συνάρτηση πρέπει να χρησιμοποιεί τη διαδικασία βαθμολόγησης(grader) `tryCombination()` για να προσδιορίσει την ορθή θέση κάθε διακόπτη και την πόρτα με την οποία είναι συνδεδεμένος και πρέπει να καλεί την `answer()` όταν προσδιορίσει αυτή την πληροφορία.

Παράμετροι

- `N`: Υποδηλώνει πόσοι διακόπτες και πόρτες υπάρχουν στη σπηλιά.

Παράδειγμα

Υποθέστε ότι οι πόρτες και οι διακόπτες έχουν διάταξη σύμφωνα με την πιο πάνω εικόνα:

Function Call	Returns	Επεξήγηση
`tryCombination([1, 0, 1, 1])`	`1`	Αυτό αντιστοιχεί στην εικόνα. Οι διακόπτες 0, 2 και 3 είναι κάτω, ενώ ο διακόπτης 1 είναι πάνω. Η συνάρτηση επιστρέφει `1`, που δείχνει ότι η πόρτα 1 είναι η πρώτη από τα αριστερά που είναι κλειστή.
`tryCombination([0, 1, 1, 0])`	`3`	Οι πόρτες 0, 1 και 2 είναι όλες ανοικτές ενώ η πόρτα 3 είναι κλειστή.
`tryCombination([1, 1, 1, 0])`	`-1`	Μετακινώντας το διακόπτη 0 κάτω έχουμε ως αποτέλεσμα όλες οι πόρτες να είναι ανοικτές, πράγμα που δηλώνεται με την επιστροφή της τιμής `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program exits)*	Υποθέτουμε ότι ο ορθός συνδυασμός είναι `[1, 1, 1, 0]`, και ότι οι διακόπτες 0, 1, 2 και 3 συνδέονται με τις πόρτες 3, 1, 0 και 2 αντίστοιχα.

Περιορισμοί

- Περιορισμός χρόνου: 2 seconds
- Περιορισμός Μνήμης: 32 MB (MiB)
- $1 \le N \le 5,000$

Υποπρόβλημα

Υποπρόβλημα	Μονάδες	Πρόσθετοι περιορισμοί εισόδου
1	12	Για κάθε `i`, ο διακόπτης `i` είναι συνδεδεμένος με την πόρτα `i`. Εσείς πρέπει απλώς να εντοπίσετε το σωστό συνδιασμό.
2	13	Ο σωστός συνδιασμός θα είναι πάντα `[0, 0, 0, ..., 0]`. Εσείς θα πρέπει απλώς να εντοπίσετε ποιός διακόπτης συνδέεται με ποιά πόρτα.
3	21	$N \le 100$
4	30	$N \le 2,000$
5	24	*(None)*

Πειραματισμός

Το πρόγραμμα αξιολόγησης (grader) πάνω στον υπολογιστή σας θα δέχεται είσοδο από το αρχείο `cave.in`, το οποίο πρέπει να έχει την εξής μορφοποίηση:

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

Εδώ `N` είναι ο αριθμός από πόρτες και διακόπτες, `S[i]` είναι η σωστή θέση για κάθε διακόπτη `i`, και `D[i]` είναι η πόρτα με την οποία ειναι συνδεδεμένος ο διακόπτης `i`.

Έτσι, το πιο πάνω παράδειγμα θα πρέπει να δοθεί με την ακόλουθη μορφοποιηση:

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

Σημειώσεις για τις γλώσσες

C/C++	Πρέπει να συμπεριλάβετε `#include "cave.h"`.
Pascal	Πρέπει να ορίσετε `unit Cave`, και να εισάξετε τις ρουτίνες του grader μέσω `uses GraderHelpLib`. Ολοι οι πίνακες αριθμούνται αρχίζοντας από `0` (not `1`).

Για παραδείγματα, δέστε τα πρότυπα λύσεων στον υπολογιστή σας.