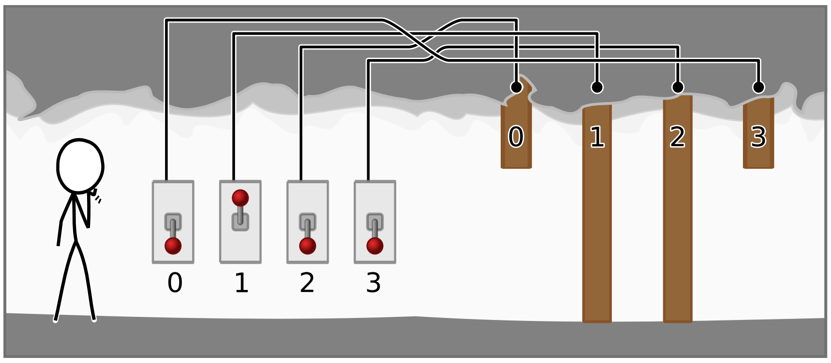
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

English — 1.0

While lost on the long walk from the college to the UQ Centre, you have stumbled across the entrance to a secret cave system running deep under the university. The entrance is blocked by a security system consisting of `N` consecutive doors, each door behind the previous; and `N` switches, with each switch connected to a different door.



The doors are numbered `0, 1, …, (N - 1)` in order, with door 0 being closest to you. The switches are also numbered `0, 1, …, (N - 1)`, though you do not know which switch is connected to which door.

The switches are all located at the entrance to the cave. Each switch can either be in an *up* or *down* position. Only one of these positions is correct for each switch. If a switch is in the correct position then the door it is connected to will be open, and if the switch is in the incorrect position then the door it is connected to will be closed. The correct position may be different for different switches, and you do not know which positions are the correct ones.

You would like to understand this security system. To do this, you can set the switches to any combination, and then walk into the cave to see which is the first closed door. Doors are not transparent: once you encounter the first closed door, you cannot see any of the doors behind it.

You have time to try `70,000` combinations of switches, but no more. Your task is to determine the correct position for each switch, and also which door each switch is connected to.

Implementation

You should submit a file that implements the procedure `exploreCave()`. This may call the grader function `tryCombination()` up to 70,000 times, and must finish by calling the grader procedure `answer()`. These functions and procedures are described below.

Grader Function: `tryCombination()`

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

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

Description

The grader will provide this function. It allows you to try a combination of switches, and then enter the cave to determine the first closed door. If all doors are open, the function will return `-1`. This function runs in `O(N)` time; that is, the running time is at worst proportional to `N`.

This function may be called at most `70,000` times.

Parameters

- `S`: An array of length `N`, indicating the position of each switch. The element `S[i]` corresponds to switch `i`. A value of `0` indicates that the switch is up, and a value of `1` indicates that the switch is down.
- *Returns*: The number of the first door that is closed, or `-1` if all doors are open.

Grader Procedure: **`answer()`**

C/C++

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

Pascal

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

Description

Call this procedure when you have identified the combination of switches to open all doors, and the door to which each switch is connected.

Parameters

- `S` : An array of length `N`, indicating the correct position of each switch. The format matches that of the function `tryCombination()` described above.
- `D`: An array of length `N`, indicating the door each switch is connected to. Specifically, element `D[i]` should contain the door number that switch `i` is connected to.
- *Returns*: This procedure does not return, but will cause the program to exit.

Your Procedure: **`exploreCave()`**

C/C++

```
void exploreCave(int N);
```

Pascal

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

Description

Your submission must implement this procedure.

This function should use the grader routine `tryCombination()` to determine the correct position for each switch and the door each switch is connected to, and must call `answer()` once it has determined this information.

Parameters

- `N` : The number of switches and doors in the cave.

Sample Session

Suppose the doors and switches are arranged as in the picture above:

Function Call	Returns	Explanation
`tryCombination([1, 0, 1, 1])`	`1`	This corresponds to the picture. Switches 0, 2 and 3 are down, while switch 1 is up. The function returns `1`, indicating that door 1 is the first door from the left that is closed.
`tryCombination([0, 1, 1, 0])`	`3`	Doors 0, 1 and 2 are all opened, while door 3 is closed.
`tryCombination([1, 1, 1, 0])`	`-1`	Moving switch 0 down causes all doors to be opened, indicated by the return value of `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program exits)*	We guess that the correct combination is `[1, 1, 1, 0]`, and that switches 0, 1, 2 and 3 connect to doors 3, 1, 0 and 2 respectively.

Constraints

- Time limit: 2 seconds
- Memory limit: 32 MiB
- $1 \le N \le 5{,}000$

Subtasks

Subtask	Points	Additional Input Constraints
1	12	For each `i`, switch `i` is connected to door `i`. Your task is simply to determine the correct combination.
2	13	The correct combination will always be `[0, 0, 0, …, 0]`. Your task is simply to determine which switch connects to which door.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(None)*

Experimentation

The sample grader on your computer will read input from the file `cave.in`, which must be in the following format:

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

Here `N` is the number of doors and switches, `S[i]` is the correct position for switch `i`, and `D[i]` is the door that switch `i` is connected to.

For instance, the example above would be provided in the following format:

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

Language Notes

C/C++	You must `#include "cave.h"`.
Pascal	You must define the `unit Cave`, and you must also import the grader routines via `uses GraderHelpLib`. All arrays are numbered beginning at `0` (not `1`).

See the solution templates on your machine for examples.