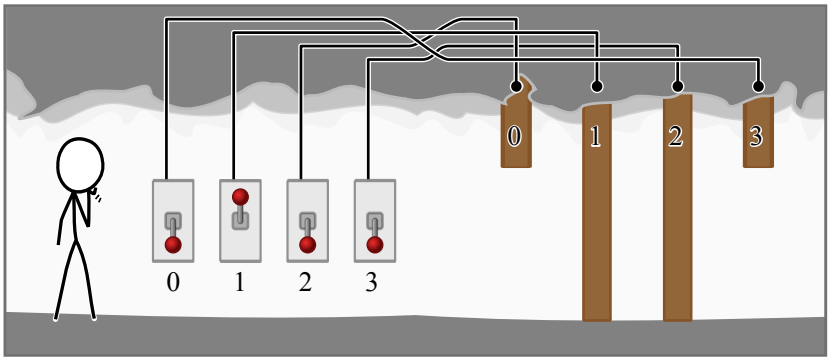
International Olympiad in Informatics 2013
6–13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Korean — 1.0

기숙사에서 UQ 센터로 가는 도중 길을 잃은 당신은 우연히 대학 지하의 비밀 동굴 입구를 찾게 되었다. 입구는 N 개의 연속된 문으로 이루어진 경비 시스템으로 막혀 있는데, 각 문은 차례대로 놓여 있고, 총 N 개의 스위치 중 하나와 연결되어 있다. 각 스위치는 정확히 하나의 문에만 연결되어 있다.



문은 차례대로 $0, 1, \cdots, (N-1)$ 로 번호가 매겨져 있고, 0번 문이 가장 앞에 있다. 스위치도 $0, 1, \cdots, (N-1)$ 로 번호가 매겨져 있지만, 어느 스위치가 어느 문에 연결되어 있는지는 주어지지 않는다.

모든 스위치는 동굴 입구에 있다. 각각의 스위치는 위나 아래로 설정될 수 있다. 각 스위치마다 이 둘 중 하나만 정답이며, 스위치를 정답 위치에 놓으면 이 스위치와 연결된 문이 열린다. 만약 스위치를 오답 위치에 놓으면 이 스위치와 연결된 문은 닫힌다. 스위치마다 정답은 다를 수 있으며, 어느 위치가 정답인지는 주어지지 않는다.

당신은 이 경비 시스템을 이해하고 싶다. 이를 위해서, 스위치를 임의의 조합으로 설정한 후 동굴 안으로 걸어가서 처음으로 닫혀 있는 문이 무엇인지를 알 수 있다. 문은 투명하지 않기 때문에, 첫 번째 닫혀 있는 문 이후의 상태는 알 수 없다.

스위치 조합은 최대 $70{,}000$ 번 시도해볼 수 있다. 당신의 임무는 각 스위치의 정답 위치를 찾고, 또 어느 스위치가 어느 문에 연결되어 있는지를 알아내는 것이다.

구현

다음 조건을 만족하는 함수 `exploreCave()`를 구현한 파일을 제출해야 한다. 이 함수는 그레이더 함수 `tryCombination()`을 최대 70,000번 호출할 수 있고, 그레이더 함수 `answer()`를 호출하는 것을 마지막으로 종료해야 한다. 이 함수들은 다음과 같이 정의된다.

그레이더 함수: **`tryCombination()`**

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

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

설명

이 함수는 그레이더에 포함되어 있다. 이 함수는 스위치 조합을 테스트해보고, 그리고 동굴에 들어가서 처음으로 닫혀 있는 문의 번호를 알려준다. 만약 모든 문이 열려 있다면 `-1`을 리턴한다. 이 함수는 `O(N)` 시간에 동작한다. 즉, 실행 시간은 최악의 경우에도 `N`에 비례한다.

이 함수는 최대 `70,000`번 호출될 수 있다.

파라미터

- `S`: 각 스위치의 상태를 나타내는 길이 `N`인 배열. `S[i]`는 `i`번 스위치에 해당한다. 이 값이 `0`이면 이 스위치가 위로 되어 있음을, `1`이면 이 스위치가 아래로 되어 있음을 나타낸다.
- 리턴값: 첫 번째 닫힌 문의 번호이며, 모든 문이 열려 있으면 `-1`를 리턴한다.

그레이더 함수: **answer()**

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

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

설명

모든 문을 열 수 있는 스위치의 조합과, 각 스위치가 어느 문에 연결되어 있는지를 알아내면 이 함수를 호출하시오.

파라미터

- `S`: 길이 `N`의 배열로, 각 스위치의 정답 위치를 나타낸다. 포맷은 위에서 설명한 `tryCombination()` 함수와 같다.
- `D`: 길이 `N`의 배열로, 각 스위치가 어느 문에 연결되어 있는지를 나타낸다. 보다 자세하게, `D[i]`는 스위치 `i`가 연결된 문의 번호를 저장한다.
- 리턴값: 이 함수는 리턴값이 없지만, 호출되면 프로그램이 종료하게 된다.

구현해야 하는 함수: **exploreCave()**

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

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

설명

이 함수를 구현하여 제출해야 한다.

이 함수는 각 스위치의 정답 위치와 각 스위치가 연결된 문을 알아내기 위해서 그레이더 함수 `tryCombination()`를 이용해야 하며, 이 정보를 알아내면 `answer()` 함수를 호출해야 한다.

파라미터

- `N`: 동굴에 있는 스위치와 문의 수

예제 세션

문과 스위치가 위 그림처럼 주어졌다고 하자.

Function Call	Returns	Explanation
`tryCombination([1, 0, 1, 1])`	`1`	앞의 그림에 해당한다. 스위치 0, 2, 3은 아래로 되어 있고 1은 위로 되어 있다. 이 함수의 리턴값은 `1`인데, 이는 왼쪽부터 봤을 때 1번 문이 닫혀 있는 첫 번째 문이라는 뜻이다.
`tryCombination([0, 1, 1, 0])`	`3`	문 0, 1, 2가 모두 열려있고, 3이 닫혀 있다.
`tryCombination([1, 1, 1, 0])`	`-1`	스위치 0을 아래로 내리면 모든 문이 열리게 되어 리턴값이 `-1`이 된다.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	(프로그램 종료)	우리가 알아낸 스위치의 정확한 조합은 `[1, 1, 1, 0]`이고, 스위치 0, 1, 2, 3은 각각 문 3, 1, 0, 2와 연결되어 있다.

제약 조건

- 시간 제한 : 2초
- 메모리 제한 : 32MB
- `1 ≤ N ≤ 5,000`

서브태스크

서브태스크	배점	추가적인 입력 제한 사항
1	12	각 스위치 `i`는 문 `i`와 연결되어 있다. 당신은 올바른 스위치 조합을 찾기만 하면 된다.
2	13	올바른 스위치 조합은 항상 `[0, 0, 0, …, 0]`이다. 당신은 어느 스위치가 어느 문에 연결되어 있는지만 알아내면 된다.
3	21	`N ≤ 100`
4	30	`N ≤ 2,000`
5	24	(없음)

테스트용 입력 형식

당신의 컴퓨터에 있는 샘플 그레이더는 입력을 파일 `cave.in`에서 읽어들이는데, 포맷은 다음과 같아야 한다.

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

여기서 `N`은 문과 스위치의 수이다. `S[i]`는 스위치 `i`의 정답 위치이며, `D[i]`는 스위치 `i`가 연결된 문이다.

예를 들어, 위의 예시는 다음과 같은 형식을 따라야 한다:

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

프로그래밍 언어 유의사항

C/C++	`#include "cave.h"`를 해야 한다.
Pascal	`unit Cave`을 정의해야 하며, 또 `uses GraderHelpLib`를 사용하여 그레이더 루틴을 불러와야 한다. 모든 배열은 `0`에서 시작한다 (`1`이 아님).

예시를 위하여 컴퓨터에 있는 솔루션 템플릿을 참조하시오.