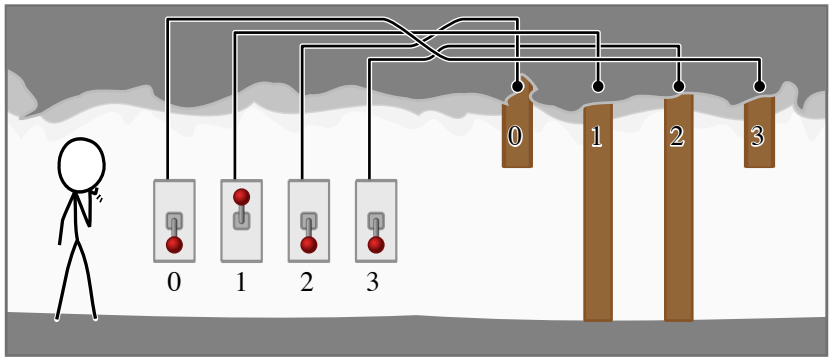
International Olympiad in Informatics
2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Japanese — 1.0

学寮 (college) からUQセンターに向かって歩いている間に迷ってしまったあなたは，この大学の地下深くを走る秘密の洞窟への入り口を偶然見つけてしまった．その入り口は N 個の連続したドアと N 個のスイッチからなるセキュリティーシステムで阻まれている．各ドアは一つ前のドアの奥にあり，各スイッチは異なるドアに接続されている．



ドアはあなたに近いほうから順に 0, 1, ⋯, (N - 1) と番号がつけられている．スイッチも 0, 1, ⋯, (N - 1) と番号がつけられているが，どのスイッチがどのドアに接続されているかはわからない．

すべてのスイッチは洞窟への入り口に設置されている．各スイッチは **上 (up)** または **下 (down)** のどちらかの状態にすることができる．各スイッチについて，一方の状態だけが正しく，もう一方の状態は正しくない．スイッチが正しい状態にある場合は，それに接続されているドアは開き，スイッチが正しくない状態にある場合は，それに接続されているドアは閉じる．どちらの状態が正しいかはスイッチによって異なるかもしれず，どちらが正しいかはわからない．

あなたはこのセキュリティーシステムを理解したいと思っている．そのために，あなたはスイッチの状態を任意の組合せに設定して，洞窟の中へ歩いて行くことができる．そして，最初の閉じているドアがどれであるかを確認することができる．ドアは透明ではない．つまり，あなたは最初の閉じているドアに遭遇してしまったら，その奥のどのドアも見ることはできない．

あなたにはスイッチの状態の組合せを 70,000 回試す時間があるが，それより多く試すことはできない．あなたの課題は，各スイッチについての正しい状態を決定することと，どのスイッチがどのドアに接続されているかを決定することである．

実装 (Implementation)

あなたは，プロシージャー `exploreCave()` を実装し，それを含む 1 つのファイルを提出しなければならない．このプロシージャーは採点プログラムにより提供される関数 `tryCombination()` を最大 70,000 回呼び出してよく，またこのプロシージャーは採点プログラムにより提供されるプロシージャー `answer()` を呼び出すことによって終了しなければならない．これらの関数およびプロシージャーの仕様は以下のとおりである．

採点プログラムの関数 (Grader Function): `tryCombination()`

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

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

説明 (Description)

採点プログラムがこの関数を提供する．この関数を使うことで，スイッチの状態の組合せを試し，最初の閉じているドアを見つけるために洞窟に入ることができる．すべてのドアが開いている場合は，この関数は `-1` を返す．この関数は O(N) 時間で動作する．つまり，この関数は `N` に比例する時間以下で実行できる．

この関数は最大 70,000 回呼び出すことができる．

引数と戻り値 (Parameters)

- `S`: 各スイッチの状態を表す長さ `N` の配列．要素 `S[i]` は スイッチ `i` に対応する．値が `0` のときはそのスイッチが上 (up) の状態であることを表し，値が `1` のときはそのスイッチが 下 (down) の状態であることを表す．
- **Returns** (戻り値): 最初の閉じているドアの番号．ただし，すべてのドアが開いている場合は `-1` を返すこと．

採点プログラムのプロシージャー (Grader Procedure): `answer()`

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

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

説明 (Description)

各スイッチの正しい状態と，各スイッチが接続されているドアが特定できた時に，このプロシージャーを呼ぶこと．

引数と戻り値 (Parameters)

- `S`：各スイッチの正しい状態を表す長さ `N` の配列．この引数の形式は上で説明されている関数 `tryCombination` の引数 `S` と同じである．
- `D`：各スイッチが接続されているドアを表す長さ `N` の配列．具体的には，要素 `D[i]` はスイッチ `i` が接続されているドアの番号でなければならない．
- **Returns** (戻り値): このプロシージャーは戻り値をもたない．ただし，このプロシージャーはプログラムを終了させる．

あなたのプロシージャー (Your Procedure): `exploreCave()`

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

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

説明 (Description)

あなたの提出は，このプロシージャーを実装していること．

このプロシージャーは，採点プログラムのルーチン `tryCombination()` を用いて各スイッチの正しい状態と各スイッチが接続されているドアを決定し，この情報を確定したら `answer()` を呼ばなければならない．

引数 (Parameters)

- `N`：洞窟内にあるスイッチとドアの個数．

やりとりの例 (Sample Session)

上の図に示されている通りにドアとスイッチが配置されているとする：

関数呼び出し	戻り値	説明
`tryCombination([1, 0, 1, 1])`	`1`	これは図に対応する．スイッチ 0, 2, 3 の状態は下であり，スイッチ 1 の状態は上である．最初に閉じているドアはドア 1 なので，この関数は `1` を返す．
`tryCombination([0, 1, 1, 0])`	`3`	ドア 0, 1, 2 は全て開いていて，ドア 3 は閉じている．
`tryCombination([1, 1, 1, 0])`	`-1`	スイッチ 0 の状態を下にすることで全てのドアが開くので，この関数は `-1` を返す．
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	**(プログラムは終了する)**	スイッチの正しい組合せは `[1, 1, 1, 0]` で，スイッチ 0, 1, 2, 3 は順にドア 3, 1, 0, 2 に接続されている．

制限 (Constraints)

- 時間制限：2 秒
- メモリ制限：32 MiB
- `1 ≤ N ≤ 5,000`

小課題 (Subtasks)

小課題	得点	入力に関する追加の制約
1	12	すべての `i` について，スイッチ `i` はドア `i` に接続されている．あなたの課題は，正しいスイッチの状態の組合せを決定することだけである．
2	13	正しいスイッチの状態の組合せは常に `[0, 0, 0, …, 0]` である．あなたの課題は，どのスイッチがどのドアに接続されているかを決定することだけである．
3	21	`N ≤ 100`
4	30	`N ≤ 2,000`
5	24	**(なし)**

試行 (Experimentation)

与えられる採点プログラムのサンプルは，ファイル `cave.in` から入力を読み込む．このファイルは次のフォーマットで書かれていなければならない：

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

ここで，`N` はドアとスイッチの数であり，`S[i]` はスイッチ `i` の正しい状態であり，`D[i]` はスイッチ `i` が接続されているドアの番号である．

例えば，上の例は次のフォーマットで与えられる：

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

言語に関する注意 (Language Notes)

C/C++ `#include "cave.h"` を行うこと．

Pascal `unit Cave` を定義すること．`uses GraderHelpLib` により採点プログラムのルーチンをインポートすること．すべての配列は `0` から始まる (`1` からではない)．

実装例については，あなたの計算機上に与えられている解法テンプレートを参照すること．