Wonders of the world

New wonders of the world have to be selected and the politically correct editor of the Siruseri Times has decided to make the exercise inclusive rather than exclusive. Instead of limiting himself to 7 wonders, he has decided to publish a list of what he considers to be the N wonders of the modern world, for some $1 \leq N \leq 2012$.

Recognizing that such a long list would bore his readers to death, he has decided to restore some interest in his list by ranking the wonders from 1 to N. To determine the rankings, he has set up a website where readers can write in their preferences. Since nobody would have the patience to list out all N wonders in order of preference, the website allows readers to rank pairs of wonders by making choices of the form "I prefer x to y".

At the end of this exercise, the Internet votes determine for each pair x and y in the list of N wonders, whether x is preferred to y or y is preferred to x. Too late, the editor realizes that it is possible that the rankings could be circular—for instance, x is preferred to y and y is preferred to z, but z is preferred to x.

He tries to salvage the situation by extracting a sequence $[x_1, x_2, \ldots, x_N]$ in which, for each $i \in \{1, 2, \ldots, N-1\}$, x_i is preferred to x_{i+1}, as voted on the website. In the example above, for instance, he can rank x, y and z as $[x, y, z]$, ignoring the fact that z is preferred to x.

Having learned some mathematics in his youth, he is able to convince himself that such a sequence can always be constructed, by induction. However, all his data is on the webserver and the site is so popular that the server is crawling and it takes a long time to recover the answer for each query of the form "Is x preferred to y?" He plans to publish the list in next weekend's magazine supplement and he calculates that he can only make $50,000$ queries before the publication deadline.

Your task is to help the editor compile the sequence he requires. For simplicity, you may assume that the wonders are named $\{1, 2, \ldots, N\}$. The ranking for each pair of wonders is obtained by calling a library function `prefer(i,j)` that returns `1` if the votes say that wonder `i` is preferred to wonder `j` and `0` otherwise. To use the library, you must first call the function `init()` that initializes the library. After making a sequence of calls to `prefer` to determine a final sequence that meets the requirements, you should store your solution in an array of size N, from position 0 to position $N-1$. The solution is reported back via a library function `solution` that takes the array with the answer as its argument. Remember that you can make at most $50,000$ calls to the function `prefer`. Full details on how to use the library functions are given at the end of the problem statement.

Input and output

Your program must not perform any input or output. Instead it will interact with a library `wonderlib`. The library contains three functions:

- `int init()`

  This function should be called *first*, at the beginning of the program, to initialize the library. The value it returns is N, the number of wonders.

- `bool prefer(int i, int j)`

  This function returns `true` if the voters prefer `i` to `j` and `false` otherwise. You are only permitted $50,000$ calls to the function `prefer`. If you make more calls, your program will be terminated and an error message "`You have exceeded the maximum number of queries allowed.`" will be printed. `i` and `j` must be distinct.

- `void solution(const int* a)`

  When you have computed a sequence that meets the editor's requirements, you should store it in an array of size at least N (size 2012 will suffice), with the wonders arranged in order from position 0 to position $N-1$. Call the function `solution`, passing the array with your answer as argument. This will terminate your program. Note that there may be more than one sequence that meets the editor's requirements. It is sufficient to identify any one.

Experimentation

This section describes how the library works, so that you can experiment with it yourself. The library reads the input from the standard input. The first line of the input is a single integer N, the number of wonders in the list. (This is the number returned when you call `init()`.)
The next N lines each contain N numbers, either 0 or 1. For $i, j \in \{1, 2, \ldots, N\}$, entry j in line i+1 is 1 if i is preferred to j and 0 otherwise. Clearly, entry j in line i+1 should be 1 if and only if entry i in line j+1 is 0. For each i, entry i in line i+1 is ignored, but it should be either 0 or 1.
To experiment with your program, you can create your own input in the format described above. After you call `solution()`, the library will write a message to the standard output indicating whether your solution was correct.

Using the library

Add the contents of the file `wonderlib.h` to the beginning of your C++ program[1]. This file declares the following functions for you to use:

```
int init();
bool prefer(int, int);
void solution(const int *);
```

To compile your program, use the following, if your code is in a file `wonder.cpp`:

```
g++ -W -Wall -O2 wonder.cpp wonderlib.o -o wonder
```

Constraints on execution

Your program should identify a final ranking of the N wonders by making at most $50,000$ calls to the function `prefer`.

- Subtask 1 (30 marks) : $1 \leq N \leq 300$.
- Subtask 2 (70 marks) : $1 \leq N \leq 2012$.

Limits

- *Time limit:* 5 s
- *Memory limit:* 128 MB

[1]Due to some limitations of the grader, a usual `#include` will not suffice.