Belt Conveyor

In the factory of JOI Co., Ltd., there are N tables, numbered from 0 to $N-1$. In the factory, there are $N-1$ belt conveyors, numbered from 0 to $N-2$. The belt conveyor i ($0 \leq i \leq N-2$) connects the table A_i and the table B_i. It transports products from one table to the other table. However, we cannot see the direction of transportation. If we ignore the directions of the belt conveyors, every pair of tables is connected by a number of belt conveyors.

IOI-kun is the director of the factory. Since he forgets the direction of transportation of every belt conveyor, he will perform the following sequential operations several times.

1. He chooses a number of belt conveyors, and reverses the directions of transportation of the chosen belt conveyors.
2. He chooses a number of tables, and puts a product on each chosen table.
3. For every table where a product is put, one of the following happens simultaneously.
   - If there is no belt conveyor transporting products from it, nothing happens.
   - If there are belt conveyors transporting products from it, the product on the table is transported by one of such belt conveyors. The product stops at the destination of the belt conveyor. The product will not move anymore.
4. IOI-kun confirms whether there are one or more products on each table. If there are products on a table, IOI-kun takes all of them.
5. For every belt conveyor whose direction is reversed in the operation 1., IOI-kun reverts its direction. Its direction becomes the original direction.

IOI-kun wants to specify the original direction of every belt conveyor by performing the above sequential operations at most 30 times.

Write a program which, given information of the tables connected by the belt conveyors, implements IOI-kun's strategy to specify the original direction of every belt conveyor by performing the above sequential operations at most 30 times.

Implementation Details

You need to submit one file.

The file is `conveyor.cpp`. It should implement the following function. The program should include `conveyor.h` using the preprocessing directive `#include`.

In `conveyor.cpp`, the following function should be implemented.

- `void Solve(int N, std::vector<int> A, std::vector<int> B)`

  This function is called only once for each test case.
  - The parameter `N` is the number of tables N.
  - The parameters `A`, `B` are arrays of length $N-1$, describing the tables connected by the belt conveyors.

  Your program can call the following function.

  ★ `std::vector<int> Query(std::vector<int> x, std::vector<int> y)`

  Using this function, IOI-kun performs the operations in the factory.
  - The parameter `x` is an array of length $N-1$. For $0 \leq$ `i` $\leq N-2$, IOI-kun reverses the direction of the belt conveyor `i` if `x[i]` $=$ `1`, and does not reverses the direction of the belt conveyor `i` if `x[i]` $=$ `0`.
  - The parameter `y` is an array of length N. For $0 \leq$ `j` $\leq N-1$, IOI-kun puts a product on the table `j` if `y[j]` $=$ `1`, and does not put a product on the table `j` if `y[j]` $=$ `0`.
  - Let `z` be the return value of this function. It is an array of length N. For $0 \leq$ `j` $\leq N-1$, there are one or more products on the table `j` if `z[j]` $=$ `1`, and there is no product on the table `j` if `z[j]` $=$ `0`.
  - The length of the array `x` should be equal to $N-1$. If this condition is not satisfied, your program is judged as **Wrong Answer [1]**.
  - Every element of the array `x` should be `0` or `1`. If this condition is not satisfied, your program is judged as **Wrong Answer [2]**.
  - The length of the array `y` should be equal to N. If this condition is not satisfied, your program is judged as **Wrong Answer [3]**.
  - Every element of the array `y` should be `0` or `1`. If this condition is not satisfied, your program is judged as **Wrong Answer [4]**.
  - The function `Query` should not be called more than 30 times. If it is called more than 30 times, your program is judged as **Wrong Answer [5]**.

  ★ `void Answer(std::vector<int> a)`

  Using this function, IOI-kun reports the original direction of each belt conveyor.
  - The parameter `a` is an array of length $N-1$. For $0 \leq$ `i` $\leq N-2$, the belt conveyor `i` transports products from A_i to B_i if `a[i]` $=$ `0`, and it transports products from B_i to A_i if `a[i]` $=$ `1`.
  - The length of the array `a` should be equal to $N-1$. If this condition is not satisfied, your program is judged as **Wrong Answer [6]**.
  - Every element of the array `a` should be `0` or `1`. If this condition is not satisfied, your program is judged as **Wrong Answer [7]**.
  - If IOI-kun reports wrong direction of a belt conveyor, your program is judged as **Wrong An-**

swer [8].

- ◇ The function `Answer` should be called exactly once. If the function `Answer` is called more than once, your program is judged as **Wrong Answer [9]**. When the function `Solve` terminates, if the function `Answer` is not called, your program is judged as **Wrong Answer [10]**.

Important Notices

- Your program can implement other functions for internal use, or use global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `conveyor.cpp`, `conveyor.h` in the same directory, and run the following command to compile your programs. Instead, you may run `compile.sh` contained in the archive file.

```
g++ -std=gnu++17 -O2 -o grader grader.cpp conveyor.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input.

N
$A_0\ A_1\ \cdots\ A_{N-2}$
$B_0\ B_1\ \cdots\ B_{N-2}$
$C_0\ C_1\ \cdots\ C_{N-2}$

For $0 \leq i \leq N-2$, we have $C_i = 0$ if the belt conveyor i transports products from the table A_i to the table B_i.

Otherwise, we have $C_i = 1$.

Output of the Sample Grader

The sample grader outputs the following information to the standard output (quotes for clarity).

- If your program is judged as correct, it reports the number of function calls to `Query` as "`Accepted: 22`".
- If your program is judged as any type of Wrong Answer, the sample grader writes its type as "`Wrong Answer [4]`".

If your program satisfies the conditions of several types of Wrong Answer, the sample grader reports only one of them.

In sample grader, among the belt conveyors transporting products from the table where a product is put, the belt conveyor transporting the product is chosen uniformly and randomly determined by pseudorandom numbers whose results do not change for different executions. In order to change the seed of pseudorandom numbers, run the sample grader with the first integer argument as follows.

```
./grader 2023
```

Notices for Grading

For some of the test cases, the actual grader is adaptive. This means the grader does not have a fixed answer in the beginning, and responds according to previous function calls to `Query`. It is guaranteed that there is at least one answer which does not contradict all the responses of the the grader.

Constraints

All the input data satisfy the following conditions.

- $0 \leq A_i \leq N - 1$ $(0 \leq i \leq N - 2)$.
- $0 \leq B_i \leq N - 1$ $(0 \leq i \leq N - 2)$.
- If we ignore the directions of the belt conveyors, every pair of tables is connected by a number of belt conveyors.

Subtasks

1. (1 point) $N = 2$.
2. (14 points) $N = 30$.
3. (10 points) $N = 100\,000$, $A_i = i$ $(0 \leq i \leq N - 2)$, $B_i = i + 1$ $(0 \leq i \leq N - 2)$.
4. (75 points) $N = 100\,000$.

Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

Sample Input 1	Sample Function Calls		
	Function Calls	Function Calls	Return Values

`3`
`0 2`
`2 1`
`1 0`
	`Solve(3, [0, 2], [2, 1])`		

		`Query([0, 0], [0, 0, 1])`	`[1, 0, 0]`
		`Query([1, 0], [1, 0, 1])`	`[0, 1, 1]`
		`Query([1, 1], [0, 0, 1])`	`[0, 0, 1]`
		`Query([0, 1], [1, 1, 1])`	`[1, 0, 1]`
		`Answer([1, 0])`	

For the first function call to `Query`, another possible return value is `[0, 1, 0]` other than `[1, 0, 0]`.

For the second function call to `Query`, the product on the table 0 is transported to the table 2 by passing through the belt conveyor 0, and stops there. Note that this product will not be transported to the table 1 by passing through the belt conveyor 1.

Note that this sample input **does not satisfy the constraints of any subtask**.

Among the files which can be obtained from the contest webpage, `sample-02.txt` satisfies the constraints of Subtask 1, and `sample-03.txt` satisfies the constraints of Subtask 2.