Library 3

After several hundred years had passed, JOI city became a ruined city. IOI-chan, an explorer, is now exploring the area where the library was built. According to the results of exploration, the following are known:

- There were a horizontal bookshelf in the library of JOI city. It had N **places** in a line to put a book, numbered from 0 to $N-1$ from left to right. Exactly one book could be placed at each place.
- There were N books in the bookshelf. The N books were numbered from 0 to $N-1$.
- The **arrangement** of books is the way to place all the N books at the N places.
- There was **correct arrangement** of the books, and the book B_i $(0 \leq i \leq N-1)$ was placed at place i in the correct arrangement. Provided that B_i is all different.

It often happens that the arrangement of the books changes. We know that in this library, the books were put back in the correct arrangement by repeating the following operation.

operation Let book x be the leftmost book among the books that the place of it differs from the place of the correct arrangement. Also let book y be the book in the current arrangement at the place where book x is placed in the correct arrangement. Swap the places of book x and book y.

Although IOI-chan found the old books of the library, she could not know the corrent arrangement. But she found an old machine which managed operations of the bookshelf of the library. If we specify the arrangement of the N books and send a query to the machine, it answers the number of operations required to put back all the books in the correct arrangement. IOI-chan wants to know the correct arrangement of the books by sending queries to the machine. Because the machine is old, she can send at most 5 000 queries to the machine.

Write a program which, given infomation of the bookshelf, specify the correct arrangement of the books by sending at most 5 000 queries.

Implementation Details

You need to submit one file.

The file is `library3.cpp`. It should implement the following function. The program should include `library3.h` using the preprocessing directive `#include`.

- `void solve(int N)`
  This function is called only once for each test case.
  - The parameter `N` is the number of books N.

In `library3.cpp`, you can call the following function

★ `int query(const std::vector<int> a)`

Using this function, IOI-chan send a queries to the machine.

- ◇ The parameter `a` is an array of length N, and it indicates the arrangement of N books to be specified in the query. It means, in the specified arrangement, book `a[i]` ($0 \leq$ `i` $\leq N-1$) must be placed at place i.
- ◇ The return value is the number of operations required to put back all the books in the correct arrangement, starting from the specified arrangement.
- ◇ The length of the parameter `a` must be N. If this condition is not met, your program will be judged as **Wrong Answer [1]**.
- ◇ Each element of the parameter `a` must be between 0 and $N-1$. If this condition is not met, your program will be judged as **Wrong Answer [2]**.
- ◇ Each element of the parameter `a` must be different from all the others. If this condition is not met, your program will be judged as **Wrong Answer [3]**.
- ◇ You must not call function `query` strictly more than 5 000 times. If it is called more than 5 000 times, your program will be judged as **Wrong Answer [4]**.

★ `void answer(std::vector<int> b)`

Your program answers the corrent arrangement of the books by calling this function.

- ◇ The parameter `b` is an array of length N, and it indicates the arrangement of N books to be specified in the answer. It means, in the answered arrangement, book `b[i]` ($0 \leq$ `i` $\leq N-1$) must be placed at place i.
- ◇ The length of the parameter `b` must be N. If this condition is not met, your program will be judged as **Wrong Answer [5]**.
- ◇ Each element of the parameter `b` must be between 0 and $N-1$. If this condition is not met, your program will be judged as **Wrong Answer [6]**.
- ◇ Each element of the parameter `b` must be different from all the others. If this condition is not met, your program will be judged as **Wrong Answer [7]**.
- ◇ The answered arrangement must be the correct arrangement. If this condition is not met, your program will be judged as **Wrong Answer [8]**.
- ◇ The function `answer` must be called exactly one time. If it is called more than 1 time, your program will be judged as **Wrong Answer [9]**. When the function `solve` terminates, if the function `answer` has not been called, your program will be judged as **Wrong Answer [10]**.

Important Notice

- 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`, `library3.cpp`, `library3.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++20 -O2 -o grader grader.cpp library3.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
$B_0\ B_1\ \cdots\ B_{N-1}$

B_i $(0 \leq i \leq N-1)$ is the book number placed at the place i in the correct arrangement.

Output of the Sample Grader

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

- If your program is judged as correct, it reports the number of calls to `query` as "`Accepted: 3000`".
- If your program is judged as any type of Wrong Answer, the sample grader writes its type as "`Wrong Answer [3]`".

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

Constraints

All the input data satisfy the following conditions.

- $2 \leq N \leq 500$.
- $0 \leq B_i \leq N - 1$ $(0 \leq i \leq N - 1)$.
- $B_i \neq B_j$ $(0 \leq i < j \leq N - 1)$.
- Given values are all integers.

Subtasks

1. (2 points) $N \leq 6$.
2. (19 points) $N \leq 100$.
3. (79 points) No additional constraints.

Sample Communication

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

Sample Input 1	Sample Function Calls		
	Call	Call	Return value
4 2 0 3 1	solve(4)		
		query([0, 1, 2, 3])	3
		query([1, 3, 0, 2])	2
		query([3, 0, 1, 2])	2
		query([2, 0, 3, 1])	0
		answer([2, 0, 3, 1])	

For example, the call of `query([0, 1, 2, 3])` specify the arrangement that books $0, 1, 2, 3$ are placed at places $0, 1, 2, 3$ respectively. The operation will be repeated as follows.

1. Swap the places of book 0 and book 1. Books $1, 0, 2, 3$ are placed at places $0, 1, 2, 3$ respectively.
2. Swap the places of book 1 and book 3. Books $3, 0, 2, 1$ are placed at places $0, 1, 2, 3$ respectively.
3. Swap the places of book 3 and book 2. Books $2, 0, 3, 1$ are placed at places $0, 1, 2, 3$ respectively.

Because number of operation required to put back all the books in the correct arrangement is 3, the return value is 3.

This sample input satisfies the constraints of all subtasks.