# Library

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 $N$ books in the bookshelf of the library in JOI city. The $N$ books were placed in the bookshelf in a line from left to right.
- The $N$ books were numbered from 1 to $N$. But, the order of books in the bookshelf might be different from the order of the numbers of the books.
- By a single operation, it was possible to take contiguously placed books from the bookshelf at once.

Unfortunately, IOI-chan could not find old books in the library. But she found a machine which managed operations of the bookshelf of the library. If we specify one or more than one books by their numbers and send a query to the machine, it answers the minimum number of operations required to take only these books from the bookshelf.

IOI-chan wants to know the order of the books in the bookshelf by sending queries to the machine. However, because the answers from the machine would be the same if the order of the $N$ books are reversed, she does not need to specify whether the books were placed from left to right or from right to left.

Because the machine is old, she can send at most 20 000 queries to the machine.

## Task

Write a program which specifies the order of the books in the bookshelf by sending at most 20 000 queries to the machine. It is not necessary to specify whether the books were placed from left to right or from right to left.

## Implementation Details

You need to submit one file.

The name of the file is `library.cpp`. It should implement the following function. The program should include `library.h`.

- `void Solve(int N)`

  For each test case, this function is called once.

  - The parameter `N` is the number of books $N$ in the bookshelf.

  Your program can call the following function.

★ `int Query(const std::vector<int>& M)`

If one or more than one books are specified by their numbers, this function returns the minimum number of operations required to take only these books from the bookshelf.

- ◇ Books taken from the bookshelf are specified by the parameter `M`, which is a vector of size $N$. For each $i$ ($1 \leq i \leq N$), if `M[i-1]` = 0, then the book $i$ is not taken from the bookshelf. If `M[i-1]` = 1, then the book $i$ is taken from the bookshelf. If the size of `M` is different from $N$, your program is considered as **Wrong Answer [1]**. For each $i$, `M[i-1]` should be equal to either 0 or 1. There should exist at least one $i$ ($1 \leq i \leq N$) with `M[i-1]` = 1. If at least one of these two conditions is not satisfied, your program is considered as **Wrong Answer [2]**. If the function `Query` is called more than 20 000 times, your program is considered as **Wrong Answer [3]**.

★ `void Answer(const std::vector<int>& res)`

Using this function, your program answers the order of the books in the bookshelf. It is not necessary to specify whether the books were placed from left to right or from right to left.

- ◇ The parameter `res` is a vector of size $N$. It describes the order of the books in the bookshelf. For each $i$ ($1 \leq i \leq N$), the $i$-th book from left in the bookshelf has number `res[i-1]`. If the size of `res` is different from $N$, your program is considered as **Wrong Answer [4]**. `res[i-1]` should be an integer between 1 and $N$, inclusive. If this condition is not satisfied, your program is considered as **Wrong Answer [5]**. Also, the integers `res[0]`, `res[1]`,..., `res[N-1]` should be different from each other. If this condition is not satisfied, your program is considered as **Wrong Answer [6]**.

When the function `Solve` terminates, if the number of calls to the function `Answer` is different from 1, your program is considered as **Wrong Answer [7]**.

If the order of the books specified by the function `Solve` is different from the order of the books in the bookshelf, your program is considered as **Wrong Answer [8]**. It is not necessary to specify whether the books were placed from left to right or from right to left.

### Important Notices

- Your program can implement other functions for internal use, or use global variables.
- Your program should not use the standard input and the standard output. Your program should not communicate with other files by any methods. But, 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 consists of one source file, which is `grader.cpp`. If your program is `library.cpp`, to test them, your put these files (`grader.cpp`, `library.cpp`) and `library.h` in the same directory, and run the following commands to compile your programs.

```
g++ -std=c++14 -O2 -o grader grader.cpp library.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.

- The first line contains the integer $N$, the number of books in the bookshelf.
- The $i$-th line ($1 \leq i \leq N$) of the following $N$ lines contains the integer $A_i$. This means the number of the $i$-th book from left in the bookshelf is $A_i$.

### Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output. (The quotation mark is not written actually.)

- If your program is considered as correct, the sample grader writes the number of calls to the function `Query` in the following form "`Accepted : 100`."
- If your program is considered as Wrong Answer, the sample grader writes its type in the following form "`Wrong Answer [1]`."

If your program is considered as several types of Wrong Answer, the sample grader reports only one of them.

## Constraints

All input data satisfy the following conditions. For the meaning of $N$ and $A_i$, see Input for the Sample Grader.

- $1 \leq N \leq 1\,000$.
- $1 \leq A_i \leq N$ ($1 \leq i \leq N$).
- $A_i \neq A_j$ ($1 \leq i < j \leq N$).

## Subtask

There are 2 subtasks. The score and additional constraints of each subtask are as follows:

### Subtask 1 [19 points]

- $N \leq 200$.

### Subtask 2 [81 points]

There are no additional constraints.

## Sample Communication

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

| Sample Input 1 | Sample Calls | | | |
|---|---|---|---|---|
| | Call | Return | Call | Return |
| 5 | `Solve(5)` | | | |
| 4 | | | `Query({1,1,1,0,0})` | |
| 2 | | | | 2 |
| 5 | | | `Answer({4,2,5,3,1})` | |
| 3 | | | | (none) |
| 1 | | | | |

In this task, it is not necessary to specify whether the books were placed from left to right or from right to left. Hence your program is considered as correct if it calls `Answer({1,3,5,2,4})` whose parameters are reversed.