# Problem L
## Leaking Santa's Secrets
### Time Limit: 1 second

It's Christmas time! While most workplaces are participating in Secret Santa gift exchanges, your workplace is hatching a more sinister plot: uncovering Santa's secrets.

Santa has a naughty-or-nice list for every human on Earth. Because its contents are so sensitive, the list is written in North-Polish, an arcane language with $N$ letters. For further security, Santa has enciphered this list with a *substitution cipher*: a permutation $H$ of the numbers $1, 2, \ldots, N$ that maps each North-Polish letter $i$ to a distinct letter $H(i)$. In such a cipher, no two letters map to the same target letter—formally, if $i \neq j$, then $H(i) \neq H(j)$—since otherwise Santa would not be able to decipher his list! (Santa can choose to map some letters to themselves, $H(i) = i$, just to be extra tricky.)

Fortunately for you, Santa's server was poorly vibe-coded and is exposed to the public Internet. You and your coworkers hope to hack into Santa's server, decipher his list, and confirm that you're all naughty! (Hackers are always naughty.)

The server was built so that Santa could quickly check his list on the go. After a user connects to the server, it prompts them to input the list of $N$ integers $H(1), H(2), \ldots, H(N)$ encoding the permutation $H$, verifies that this list is correct, and then deciphers Santa's secret list. Through months of research, you have found a side-channel timing vulnerability. Say you type in a permutation $Q$. If $H = Q$, then the server instantly grants access. Otherwise, consider a graph on $N$ vertices, and add an edge from each vertex $i$ to vertex $H(Q(i))$. You have discovered that the server's convoluted authentication algorithm will take exactly as many seconds to respond to you with an Access Denied error message as the number of connected components in the resulting graph.

For example, suppose that $N = 4$ and the cipher permutation $H$ is the following:

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $H(i)$ | 2 | 3 | 1 | 4 |

If you try to log in to the server with input `4 3 2 1`, since this permutation does not match $H$ and since the graph described above has two connected components (one containing a cycle of edges $1 \to 4 \to 2 \to 1$ and another containing just the self-loop $3 \to 3$), the server will respond with an Access Denied error message after a delay of 2 seconds.

Note that if you try to log in to the server multiple times with different inputs $Q$, it will authenticate $Q$ each time against the *same* stored $H$. It does not change $H$ in any way in response to your inputs.

Santa will notice if his server is bombarded with unauthorized requests. You've estimated that you can only make $1\,510$ login attempts before drawing too much suspicion. Can you find an efficient strategy for determining the cipher permutation?

## Interaction

This is an interactive problem. Interaction begins by reading an integer $N$ ($1 \leq N \leq 220$) from standard input, the number of letters in North-Polish. The judge is not adaptive: the hidden permutation $H$ is chosen at this time and will not change throughout the rest of the interaction.

To attempt to log in to the server, print a line with $N$ space-separated integers $Q(1), \ldots, Q(N)$, where $Q$ is a permutation of $\{1, 2, \ldots, N\}$. Then read a single integer from standard input, the amount of time in seconds it takes for the server to respond to your input.

If this delay is $0$, you have successfully found the cipher permutation $H$ and your program should terminate. If not, this delay is the number of connected components in the graph built according to the procedure described above.

You may attempt to log in at most $1\,510$ times. If you run out of attempts, your program should exit cleanly (though it will be judged incorrect for failing to decipher Santa's naughty-or-nice list).

### Notes

**Do not forget to flush the output stream after each line that you print** and to cleanly exit after the interaction is done. Please also make sure that you follow the above interaction protocol **exactly** regarding what information to print on which line of output: for example, if the protocol requires you to print a list of space-separated integers on a single line, the judge **will not** accept each integer on its own line.

If the judge receives invalid or unexpected input, it will print $-1$ and then immediately exit. Your program must detect this and cleanly exit in order to receive a Wrong Answer verdict. If your program blocks waiting for further interaction from the judge, or tries to interpret the $-1$ as the number of components, you may receive a different rejected verdict (such as Time Limit Exceeded or Runtime Error) instead of Wrong Answer.

You have been provided with a command-line tool for local testing. The tool has comments at the top explaining its use.

| Read | Sample Interaction 1 | Write |
|---|---|---|
| 3 | | |
| | | 1 2 3 |
| 1 | | |
| | | 2 1 3 |
| 2 | | |
| | | 3 1 2 |

```
0
```

This page is intentionally left blank.