

Task: ZER

Zeroes and ones



BOI 2025, Day 0. Available memory: 256 MB.

2025.04.25

binari sequeñcini a_0, \dots, a_{n-1} . elementini unknownini, but askini sumini of two distinctini elementini.
origini binari sequeñcini?

Interaction

This is an interactive task. You should write a program which finds a correct solution to the task and communicates with the interactor by reading from the standard input and writing to the standard output.

At the beginning of the interaction, your program should read an integer n ($3 \leq n \leq 200\,000$) from the standard input, denoting the length of the sequence.

Then your program should ask the questions in the following way:

- Your program should write a single line in the form of

$i \ j$

to the standard output, where i and j are distinct integers ($0 \leq i \neq j \leq n-1$). This line represents a single question concerning elements a_i and a_j .

- The response will be given as:

s

where s is an integer ($0 \leq s \leq 2$) equal to $a_i + a_j$.

Once you have determined the sequence, you should write two lines in the form of

n

$a_0 \ a_1 \ \dots \ a_{n-1}$

to the standard output, where n is the length of the sequence and a_0, a_1, \dots, a_{n-1} are its elements. **Then you should end the execution of your program. Asking further queries may result in a Wrong answer verdict.**

Keep in mind that after each query or answer you have to flush the output buffer using `cout.flush()` (or `fflush(stdout)` if using `printf`) in C++ or add the `flush = True` argument to the print command in Python. Otherwise your program may receive a Time Limit Exceeded verdict.

Your program cannot open any files or use any other resources. It can use the standard error stream for debugging purposes, but please mind that writing to this stream takes time.

C++, input/output using streams

You should include the appropriate header (`#include <iostream>`). You should end each line of the output with `std::endl`. See the following example of the initial interaction.

```
std::cin >> n;
std::cout << i << ' ' << j << std::endl;
std::cin >> sum;
```

C++, input/output using stdio

You should include the appropriate header (`#include <stdio.h>`). After outputting each line you should invoke `fflush(stdout)`. See the following example of the initial interaction.

```
scanf("%d", &n);
printf("%d %d\n", i, j);
fflush(stdout);
scanf("%d", &sum);
```

Python

Each call to the print function should have `flush = True` as an argument. See the following example of the initial interaction.

```
n = int(input())
print(f"{i} {j}", flush = True)
sum = int(input())
```

Example interaction

Action	Parameters	Output	Description
read n	–	5	$n = 5$
sum	0 1	1	$a_0 + a_1 = 1$
sum	1 2	1	$a_1 + a_2 = 1$
sum	3 4	2	$a_3 + a_4 = 2$, so $a_3 = a_4 = 1$
sum	0 3	2	$a_0 + a_3 = 2$, so $a_0 = 1$, and further $a_1 = 0$ and $a_2 = 1$
answer	5 1 0 1 1 1		Correct answer, $m = 4 \leq n = 5$ questions, 100% points.

Scoring

Subtask	Constraints	Points
1	$3 \leq n \leq 1000$	50
2	$3 \leq n \leq 200\,000$	50

The interactor **does not** need to fix the whole input sequence a_0, \dots, a_{n-1} in the very beginning of the interaction. Instead, it can modify the elements of the current sequence, as long as this is consistent with all the results returned by the queries asked so far. In other words, the interactor is adaptive.

Denoting by m the maximum number of queries asked by your program, you will receive the following percentage of the number of points for a given test:

Queries	Percentage of points
$m \leq n$	100% points for the test
$m = n + 1$	80% points for the test
$n + 1 < m \leq n^2 - n$	50% points for the test
$m > n^2 - n$	0% points for the test (Wrong answer)