Codeforces Round #580 (Div. 2)

A. Choose Two Numbers

1 second, 256 megabytes

You are given an array A, consisting of n positive integers $a_1, a_2, \ldots, a_n$, and an array B, consisting of m positive integers $b_1, b_2, \ldots, b_m$.

Choose some element a of A and some element b of B such that $a + b$ doesn't belong to A and doesn't belong to B.

For example, if $A = [2, 1, 7]$ and $B = [1, 3, 4]$, we can choose 1 from A and 4 from B, as number $5 = 1 + 4$ doesn't belong to A and doesn't belong to B. However, we can't choose 2 from A and 1 from B, as $3 = 2 + 1$ belongs to B.

It can be shown that such a pair exists. If there are multiple answers, print any.

Choose and print any such two numbers.

Input

The first line contains one integer n ($1 \le n \le 100$) — the number of elements of A.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 200$) — the elements of A.

The third line contains one integer m ($1 \le m \le 100$) — the number of elements of B.

The fourth line contains m different integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 200$) — the elements of B.

It can be shown that the answer always exists.

Output

Output two numbers a and b such that a belongs to A, b belongs to B, but $a + b$ doesn't belong to nor A neither B.

If there are multiple answers, print any.

input
1 20 2 10 20
output
20 20

input
3 3 2 2 5 1 5 7 7 9
output
3 1

input
4 1 3 5 7 4 7 5 3 1
output
1 1

In the first example, we can choose 20 from array $[20]$ and 20 from array $[10, 20]$. Number $40 = 20 + 20$ doesn't belong to any of those arrays. However, it is possible to choose 10 from the second array too.

In the second example, we can choose 3 from array $[3, 2, 2]$ and 1 from array $[1, 5, 7, 7, 9]$. Number $4 = 3 + 1$ doesn't belong to any of those arrays.

In the third example, we can choose 1 from array $[1, 3, 5, 7]$ and 1 from array $[7, 5, 3, 1]$. Number $2 = 1 + 1$ doesn't belong to any of those arrays.

B. Make Product Equal One

1 second, 256 megabytes

You are given n numbers $a_1, a_2, \ldots, a_n$. With a cost of one coin you can perform the following operation:

Choose one of these numbers and add or subtract 1 from it.

In particular, we can apply this operation to the same number several times.

We want to make the product of all these numbers equal to 1, in other words, we want $a_1 \cdot a_2 \ldots \cdot a_n = 1$.

For example, for $n = 3$ and numbers $[1, -3, 0]$ we can make product equal to 1 in 3 coins: add 1 to second element, add 1 to second element again, subtract 1 from third element, so that array becomes $[1, -1, -1]$. And $1 \cdot (-1) \cdot (-1) = 1$.

What is the minimum cost we will have to pay to do that?

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of numbers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the numbers.

Output

Output a single number — the minimal number of coins you need to pay to make the product equal to 1.

input
2 -1 1
output
2

input
4 0 0 0 0
output
4

input
5 -5 -3 5 3 0
output
13

In the first example, you can change 1 to -1 or -1 to 1 in 2 coins.

In the second example, you have to apply at least 4 operations for the product not to be 0.

In the third example, you can change -5 to -1 in 4 coins, -3 to -1 in 2 coins, 5 to 1 in 4 coins, 3 to 1 in 2 coins, 0 to 1 in 1 coin.

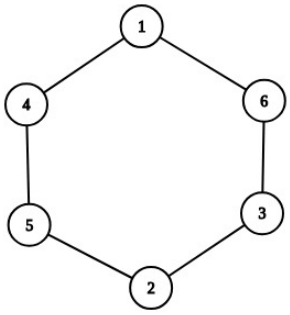
C. Almost Equal

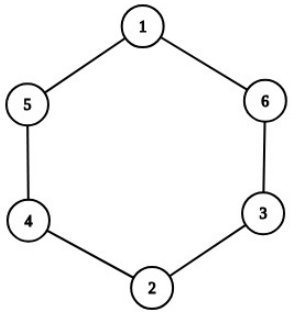
1 second, 256 megabytes

You are given integer n. You have to arrange numbers from 1 to $2n$, using each of them exactly once, on the circle, so that the following condition would be satisfied:

For every n consecutive numbers on the circle write their sum on the blackboard. Then any two of written on the blackboard $2n$ numbers differ not more than by 1.

For example, choose $n = 3$. On the left you can see an example of a valid arrangement: $1 + 4 + 5 = 10$, $4 + 5 + 2 = 11$, $5 + 2 + 3 = 10$, $2 + 3 + 6 = 11$, $3 + 6 + 1 = 10$, $6 + 1 + 4 = 11$, any two numbers differ by at most 1. On the right you can see an invalid arrangement: for example, $5 + 1 + 6 = 12$, and $3 + 2 + 4 = 9$, 9 and 12 differ more than by 1.





Input

The first and the only line contain one integer n ($1 \le n \le 10^5$).

Output

If there is no solution, output "NO" in the first line.

If there is a solution, output "YES" in the first line. In the second line output $2n$ numbers — numbers from 1 to $2n$ in the order they will stay in the circle. Each number should appear only once. If there are several solutions, you can output any of them.

input
3
output
YES 1 4 5 2 3 6

input
4
output
NO

Example from the statement is shown for the first example.

It can be proved that there is no solution in the second example.

D. Shortest Cycle

1 second, 256 megabytes

You are given n integer numbers $a_1, a_2, \ldots, a_n$. Consider graph on n nodes, in which nodes i, j ($i \neq j$) are connected if and only if, a_i AND $a_j \neq 0$, where AND denotes the bitwise AND operation.

Find the length of the shortest cycle in this graph or determine that it doesn't have cycles at all.

Input

The first line contains one integer n ($1 \le n \le 10^5$) — number of numbers.

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^{18}$).

Output

If the graph doesn't have any cycles, output -1. Else output the length of the shortest cycle.

input
4 3 6 28 9
output
4

input
5 5 12 9 16 48
output
3

input
4 1 2 4 8
output
-1

In the first example, the shortest cycle is $(9, 3, 6, 28)$.

In the second example, the shortest cycle is $(5, 12, 9)$.

The graph has no cycles in the third example.

E. Palindromic Paths

1 second, 256 megabytes

This is an interactive problem

You are given a grid $n \times n$, where n is **odd**. Rows are enumerated from 1 to n from up to down, columns are enumerated from 1 to n from left to right. Cell, standing on the intersection of row x and column y, is denoted by (x, y).

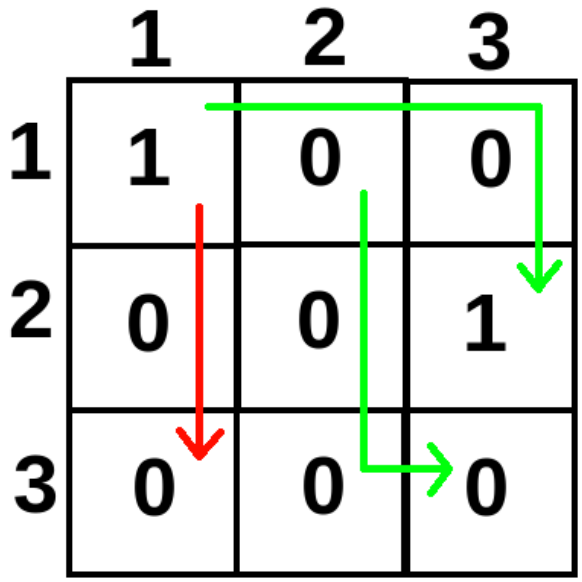
Every cell contains 0 or 1. It is known that the top-left cell contains 1, and the bottom-right cell contains 0.

We want to know numbers in all cells of the grid. To do so we can ask the following questions:

"? x_1 y_1 x_2 y_2", where $1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$, and $x_1 + y_1 + 2 \le x_2 + y_2$. In other words, we output two different cells (x_1, y_1), (x_2, y_2) of the grid such that we can get from the first to the second by moving only to the right and down, and they aren't adjacent.

As a response to such question you will be told if there exists a path between (x_1, y_1) and (x_2, y_2), going only to the right or down, numbers in cells of which form a palindrome.

For example, paths, shown in green, are palindromic, so answer for "? 1 1 2 3" and "? 1 2 3 3" would be that there exists such path. However, there is no palindromic path between $(1, 1)$ and $(3, 1)$.



Determine all cells of the grid by asking not more than n^2 questions. It can be shown that the answer always exists.

Input

The first line contains **odd** integer ($3 \le n < 50$) — the side of the grid.

Interaction

You begin the interaction by reading n.

To ask a question about cells $(x_1, y_1), (x_2, y_2)$, in a separate line output "? x_1 y_1 x_2 y_2".

Numbers in the query have to satisfy $1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$, and $x_1 + y_1 + 2 \le x_2 + y_2$. Don't forget to 'flush', to get the answer.

In response, you will receive 1, if there exists a path going from (x_1, y_1) to (x_2, y_2) only to the right or down, numbers in cells of which form a palindrome, and 0 otherwise.

In case your query is invalid or you asked more than n^2 queries, program will print -1 and will finish interaction. You will receive **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

When you determine numbers in all cells, output "!".

Then output n lines, the i-th of which is a string of length n, corresponding to numbers in the i-th row of the grid.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hack Format

To hack, use the following format.

The first line should contain a single odd integer n (side of your grid).

The i-th of n following lines should contain a string of length n corresponding to the i-th row of the grid. Top left element of the grid has to be equal to 1, bottom right has to be equal to 0.

input

```
3
0
1
0
1
1
1
1
```

output

```
? 1 1 1 3
? 1 1 2 3
? 2 1 2 3
? 3 1 3 3
? 2 2 3 3
? 1 2 3 2
? 1 2 3 3
!
100
001
000
```

F. Almost All

1 second, 256 megabytes

You are given a tree with n nodes. You have to write non-negative integers on its edges so that the following condition would be satisfied:

For every two nodes i, j, look at the path between them and count the sum of numbers on the edges of this path. Write all obtained sums on the blackboard. Then every integer from 1 to $\lfloor \frac{2n^2}{9} \rfloor$ has to be written on the blackboard at least once.

It is guaranteed that such an arrangement exists.

Input

The first line contains a single integer n ($1 \le n \le 1000$) — the number of nodes.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), meaning that there is an edge between nodes u and v. It is guaranteed that these edges form a tree.

Output

Output $n - 1$ lines, each of form u v x ($0 \le x \le 10^6$), which will mean that you wrote number x on the edge between u, v.

Set of edges (u, v) has to coincide with the set of edges of the input graph, but you can output edges in any order. You can also output ends of edges in an order different from the order in input.

input

```
3
2 3
2 1
```

output

```
3 2 1
1 2 2
```

input

```
4
2 4
2 3
2 1
```

output

```
4 2 1
3 2 2
1 2 3
```

input

```
5
1 2
1 3
1 4
2 5
```

output

```
2 1 1
5 2 1
3 1 3
4 1 6
```

In the first example, distance between nodes 1 and 2 is equal to 2, between nodes 2 and 3 to 1, between 1 and 3 to 3.

In the third example, numbers from 1 to 9 (inclusive) will be written on the blackboard, while we need just from 1 to 5 to pass the test.