Acorn Duel (acorn)

✍ This is an interactive problem. Your program must communicate with the evaluator: it should alternately write messages via standard output and read inputs from standard input.

Chippy and Nutmeg, two rival squirrels, are racing to claim acorns scattered across trees numbered $0, 1, 2, \ldots$ in a magical forest.

They begin by placing two tokens on distinct trees X and Y ($X \neq Y$). The squirrels take turns moving **one** of the two tokens to a strictly smaller-numbered tree, ensuring that both tokens never occupy the same tree. Chippy always moves first, and they continue alternating turns. The first squirrel unable to make a valid move loses the duel—along with their acorns!

Figure 1: Chippy and Nutmeg fighting over an acorn before learning this game.

Determine the winner assuming both play optimally, and help them make the best moves each turn.

☞ Among the attachments of this task you may find a template file `acorn.*` with a sample incomplete implementation.

Implementation

The first line of input contains two integers, X and Y, representing the initial positions of the two tokens.

If Chippy wins the game, print a single line containing `C`. If Nutmeg wins, print a single line containing `N`. Your program will then play as the squirrel you determined to be the winner.

- On your turn, print a single line with the new positions of the tokens, in any order, using the format: `new_X new_Y`
- The interactor will then respond with a line containing the moves made by the opposing squirrel, also in the same format.
- The game continues until one player is unable to make a valid move. At this point, your program should terminate.

After printing to `stdout`, **do not forget to print the end of line character** (`'\n'`) **and to flush the output**. Otherwise, you may receive either the `Time Limit Exceeded` or the `Memory Limit Exceeded` verdict. To flush the output, use:

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

Constraints

- $0 \le X, Y \le 100\,000$.
- $X \neq Y$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (20 points) $X, Y \le 10$.

– **Subtask 3** (20 points) $X, Y \le 100$.

– **Subtask 4** (25 points) $X, Y \le 5000$.

– **Subtask 5** (35 points) No additional limitations.

Examples

`input`	`output`

`2 3`
 	
`2 0`
	
	`N`
	
	`1 0`

input	output
4 7	
	C
	4 5
3 5	
	3 2
3 1	
	0 1

Explanation

In the **first sample case**, the tokens are initially placed on trees 2 and 3. The solution chooses to play as Nutmeg.

- The interactor, playing as Chippy, moves the second token from tree 3 to tree 0 on the first move.
- The solution (playing as Nutmeg) then moves the first token from tree 2 to tree 1.
- Now, Chippy is unable to make a valid move, so Nutmeg wins the game.

It can be proven that, in this starting configuration, Nutmeg always has a winning strategy when playing optimally.