Problem C: Confined Catching

You are playing a board game against an AI on a square grid consisting of $n \times n$ cells. You have two game pieces and the AI has one, and each piece is placed in one of the grid cells. Your goal is to "catch" the AI's piece, that is, one (or both) of your pieces has to lie in the same cell as the AI's piece after one of your turns. When this happens, you win and the game ends. You lose if you have not won after 600 turns.

Each turn, you have up to five movement options per piece: You can move a piece up, down, left, or right to an adjacent cell (if there is one) or let the piece remain in its current cell. The AI has the same options for its piece in each of its turns. Of course, you can move your pieces completely independently from one another and even have them occupy the same cell.

Your goal is simple: Win the game! You can safely assume that this is always possible.

Initial Input

Before it is your first turn, your program will receive:

- One line with an integer n ($3 \leq n \leq 100$), giving the size of the grid.
- One line with four integers x_1, y_1, x_2, y_2 ($1 \leq x_1, y_1, x_2, y_2 \leq n$), giving your pieces' initial positions.
- One line with two integers x, y ($1 \leq x, y \leq n$), giving the AI's piece's initial position.

You can safely assume that your pieces do not lie in the same cell as the AI's piece (but they may lie in the same cell as each other).

Interaction Protocol

Your submission will be interacting with a special program called the *grader*. This means that the output of your submission is sent to the grader and the output of the grader is sent to the standard input of your submission. This interaction must follow a specific protocol:

Your submission and the grader alternate writing turns, with you going first. Your turns consist of a single line with four integers x_1, y_1, x_2, y_2 to specify the locations of your pieces after your turn (in the same order as in the initial input). Similarly, the grader will send lines with two integers x, y to indicate the AI's turn. You can safely assume that the grader will only send you valid moves, and the AI will never choose to place its piece in a cell occupied by one or both of your pieces. Your submission may take at most 600 turns.

After each of your turns you should *flush* the standard output to ensure that the request is sent to the grader. For example, you can use `fflush(stdout)` in C++, `System.out.flush()` in Java and `sys.stdout.flush()` in Python.

After you send the grader a valid turn catching the AI's piece, the game ends. Your submission should then terminate with exit code `0` as usual. For convenience, the grader will send a single line containing `0 0` to signal the end of the game, which your submission may or may not read.

Your submission will be accepted if it follows the rules and protocol above and wins the game. If it sends any invalid turn, it will be judged as "Wrong Answer".

For your convenience, we have provided you with a testing tool that lets your solution interact with a simple version of the AI opponent. It is included in the ZIP archive with sample data that you can download from the DOMjudge problem overview page.

Read **Sample Interaction 1** **Write**

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

```
1 2 3 2
```

```
3 3
```

```
1 2 3 3
```

```
0 0
```