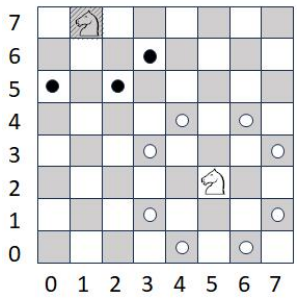
C: Knights

Source file name: `knights.c`, `knights.cpp`, `knights.java`, *or* `knights.py`
Author: Rodrigo Cardoso

KA and *KB* are knights in a generalized chessboard with dimensions $n \times n$, with $n \geq 1$. Its rows and columns are numerated $0, 1, \ldots, n-1$: the square (i, j), with $0 \leq i, j < n$, is located at the i-th row and j-th column of the chessboard.

The two knights *KA* and *KB* are positioned in squares of the board, and their goal is to wander from their initial locations to meet each other. They follow the rules of knights in chess: they can move two squares vertically and one square horizontally, or two squares horizontally and one square vertically (i.e., forming the shape of a capital L), always within the chessboard boundaries. The following figure illustrates two examples of knight moves on a typical 8×8 chessboard:



They move alternately, one move each time, and *KA* moves first. If in *KB*'s k-movement (with $k \geq 0$), *KB* may reach the square *KA* is located at, it is said that they *meet* in k movements. Note that it could be that, depending on the initial positions, *KA* and *KB* cannot meet. For instance, if on an 8×8 chessboard the initial positions of *KA* and *KB* are $(0, 0)$ and $(6, 2)$, respectively, then they can meet in 2 movements with the following sequence of movements:

$$(2, 1) \quad (5, 4) \quad (4, 2) \quad (4, 2).$$

Your task is to write a program that, given the size of the chessboard and the initial positions of *KA* and *KB*, determines the minimum number of movements needed to meet or identifies if this is impossible.

Input

The input consists of several test cases. A case is defined with a line with 5 blank-separated integer values n, a, b, c, d, $1 < n < 300$ and $0 \leq a, b, c, d < n$, where n is the number of rows and columns of the chessboard, and (a, b) and (c, d) are the initial locations for *KA* and *KB*, respectively. The end of the input is given with $n = a = b = c = d = 0$, which should not be processed.

The input must be read from standard input.

Output

For each test case, output a single line: the minimum number of moves required for *KA* and *KB* to meet whenever this is possible, and '*' otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
8 0 0 5 4 3 0 0 0 1 6 1 2 5 4 0 0 0 0 0	2 * 1