Michael The Kid receives an interesting game set from his grandparent as his birthday gift. Inside the game set box, there are n tiling blocks and each block has a form as follows:

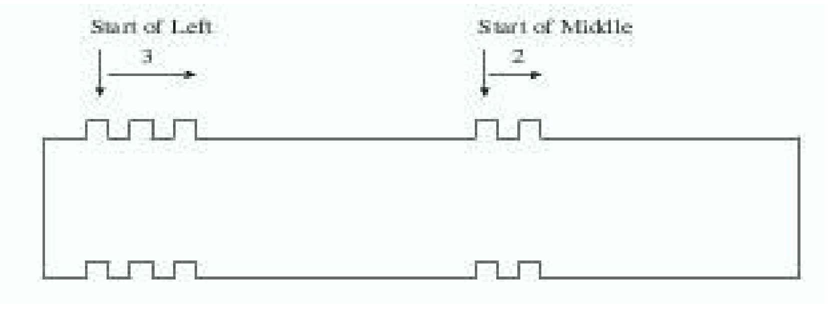

Figure 1: Michael's Tiling Block with parameters (3,2).

Each tiling block is associated with two parameters (l, m), meaning that the upper face of the block is packed with l protruding knobs on the left and m protruding knobs on the middle. Correspondingly, the bottom face of an (l, m)-block is carved with l caving dens on the left and m dens on the middle.

It is easily seen that an (l, m)-block can be tiled upon another (l, m)-block. However, this is not the only way for us to tile up the blocks. Actually, an (l, m)-block can be tiled upon another (l', m')-block if and only if $l \geq l'$ and $m \geq m'$.

Now the puzzle that Michael wants to solve is to decide what is the tallest tiling blocks he can make out of the given n blocks within his game box. In other words, you are given a collection of n blocks $B = \{b_1, b_2, \ldots, b_n\}$ and each block b_i is associated with two parameters (l_i, m_i). The objective of the problem is to decide the number of tallest tiling blocks made from B.

Input

Several sets of tiling blocks. The inputs are just a list of integers. For each set of tiling blocks, the first integer n represents the number of blocks within the game box. Following n, there will be n lines specifying parameters of blocks in B; each line contains exactly two integers, representing left and middle parameters of the i-th block, namely, l_i and m_i. In other words, a game box is just a collection of n blocks $B = \{b_1, b_2, \ldots, b_n\}$ and each block b_i is associated with two parameters (l_i, m_i).

Note that n can be as large as 10000 and l_i and m_i are in the range from 1 to 100. An integer $n = 0$ (zero) signifies the end of input.

Output

For each set of tiling blocks B, output the number of the tallest tiling blocks can be made out of B. Output a single star '*' to signify the end of outputs.

Sample Input

```
3
3 2
1 1
2 3
5
4 2
2 4
3 3
1 1
5 5
0
```

Sample Output

```
2
3
*
```