


Function Tiling Puzzle

15-Tile game is a puzzle which is combined from 15 pieces that has a number on it from 1 – 15.

The win condition of the game is to order the pieces into ascending order like the example below.

1	7	2	3
6		8	4
5	9	10	11
13	14	15	12



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

The table can be interpreted to list of lists of integers (0 representing the space). For example, the picture above can be represented by

```
[ [1,7,2,3], [6,0,8,4], [5,9,10,11], [13,14,15,12] ]
```

For your intention, there is some table that cannot be solved. For example, If you swap the position of 14 and 15 as `[[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,15,14,0]]`, this puzzle cannot be solved. (We will not proof this here)

Your task is to implement functions which help to check that the table (list of lists) can be solved or not.

This can be checked without any move by these following steps. (for table nxn)

1. flatten: change a **list of lists of ints** to **list of ints** then remove 0, for example, change `[[1,2,0], [3,5,6], [4,7,8]]` to `[1,2,3,5,6,4,7,8]`
2. inversions: find pairs of inversion which are tuples of data in the list and count the pairs that have its left value greater than its right value. For example, there are 8 numbers, so there are $8*7/2 = 28$ pairs:

```
(1,2), (1,3), (1,5), (1,6), (1,4), (1,7), (1,8), (2,3), (2,5), (2,6), (2,4), (2,7), (2,8), (3,5), (3,6), (3,4), (3,7), (3,8), (5,6), (5,4), (5,7), (5,8), (6,4), (6,7), (6,8), (4,7), (4,8), (7,8)
```

There are 2 pairs which the left number is more than the right number, in this case inversion is 2

3. the puzzle can be solved when

Number of rows of the table	Number of inversions	Row where 0 exist (The first row is 0)
odd	even	wherever
even	odd	even
	even	odd

From the example in step one, the table has 3 rows which is an odd number, and the inversions is even, so this problem is solvable.

For more detail: https://en.wikipedia.org/wiki/15_puzzle

Your task is to implement the following functions:

```
def row_number(t, e): # returns row number of t containing e
                      # (top row is row #0)

def flatten(t):       # returns a list of ints converted from list of
                      # lists of this t

def inversions(x):    # returns the number of inversions of list x

def solvable(t):      # returns True if tiling t (list of lists of ints)
                      # is solvable otherwise returns False

exec(input().strip()) # do not remove this line
```

Input

Python commands to test the functions.

Output

Result from the commands.

Example

Input (from keyboard)	Output (on screen)
<code>print(row_number([[0,8,7],[6,5,4],[3,2,1]], 0))</code>	0
<code>print(flatten([[0,8,7],[6,5,4],[3,2,1]]))</code>	[8, 7, 6, 5, 4, 3, 2, 1]
<code>print(inversions([8,7,6,5,4,3,2,1]))</code>	28
<code>print(solvable([[0,8,7],[6,5,4],[3,2,1]]))</code>	True

