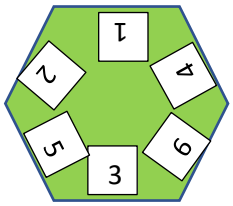
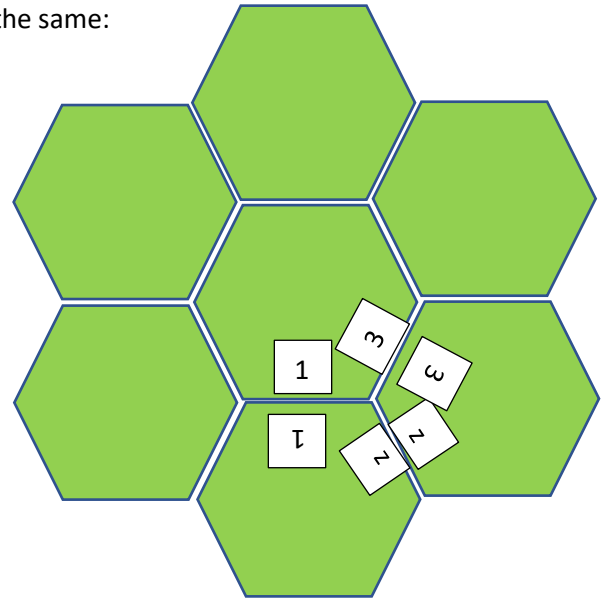
The Honeycomb Puzzle

In this puzzle, there are seven hexagons whose edges are numbered using a permutation of numbers 1 to 6. An example of such a hexagon is in the figure below:



A solution of the puzzle is a configuration of the hexagons, in which one is placed in the center and the rest around it, so that whenever two hexagons touch each other, the adjacent edges are always numbered the same:



Your must write a ***recursive***[*] methore that accepts as an input the seven *different* hexagons (i.e., each numbered with a permutation different from others), tries all possible configurations for this input, and finds and displaus a solution of the puzzle—if it is soluble—or reports that there is no solution.

Specifically, show that a solution exists for the following seven hexagons (*numbering of the edges is always counter-clockwise*):

I. (1, 2, 3, 4, 5, 6)

II. (1, 6, 4, 2, 5, 3)

III. (1, 6, 5, 4, 3, 2)

---

[*] A non-recursive solution won't be considered, and no partial credit will be given for it.

IV. (1, 4, 6, 2, 3, 5)

V. (1, 6, 5, 3, 2, 4)

VI. (1, 4, 3, 6, 5, 2)

VII. (1, 6, 2, 4, 5, 3)

You have to create a test case that would try other numberings—including those that contain the same ones, in which case your program must produce an error.

Can you try all possible hexagon combinations? If so, write a *recursive* program that generates every such a configuration and call your method to solve it. Keep a count of soluble configurations you have generated.