Problem

As punishment for being naughty, Dante has been trapped in a strange house with many rooms. The house is an $\mathbf{N} \times \mathbf{N}$ grid of rooms, with $\mathbf{N}$ odd and greater than 1. The upper left room is numbered 1, and then the other rooms are numbered 2, 3, ..., $\mathbf{N}^2$, in a clockwise spiral pattern. That is, the numbering proceeds along the top row of the grid and then makes a 90 degree turn to the right whenever a grid boundary or an already numbered room is encountered, and finishes in the central room of the grid. Because $\mathbf{N}$ is odd, there is always a room in the exact center of the house, and it is always numbered $\mathbf{N}^2$.

For example, here are the room numberings for houses with $\mathbf{N} = 3$ and $\mathbf{N} = 5$:

1	2	3
8	9	4
7	6	5

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

Dante starts off in room 1 and is trying to reach the central room (room $\mathbf{N}^2$). Throughout his journey, he can only make moves from his current room to higher-numbered, adjacent rooms. (Two rooms must share an edge — not just a corner — to be adjacent.)

Dante knows that he could walk from room to room in consecutive numerical order — i.e., if he is currently in room x, he would move to room $x+1$, and so on. This would take him exactly $\mathbf{N}^2 - 1$ moves. But Dante wants to do things his way! Specifically, he wants to reach the central room in exactly $\mathbf{K}$ moves, for some $\mathbf{K}$ strictly less than $\mathbf{N}^2 - 1$.

Dante can accomplish this by taking one or more *shortcuts*. A shortcut is a move between rooms that are not consecutively numbered.

For example, in the 5×5 house above,

- If Dante is at 1, he cannot move to 17, but he can move to 2 or to 16. The move to 2 is not a shortcut, since $1 + 1 = 2$. The move to 16 is a shortcut, since $1 + 1 \neq 16$.
- From 2, it is possible to move to 3 (not a shortcut) or to 17 (a shortcut), but not to 1, 16, or 18.
- From 24, Dante can only move to 25 (not a shortcut).
- It is not possible to move out of room 25.

As a specific example using the 5×5 house above, suppose that $\mathbf{K} = 4$. One option is for Dante to move from 1 to 2, then move from 2 to 17 (which is a shortcut), then move from 17 to 18, then move from 18 to 25 (which is another shortcut). This is illustrated below (the red arrows represent shortcuts):

Can you help Dante find a sequence of exactly $\mathbf{K}$ moves that gets him to the central room, or tell him that it is impossible?

Naive Brute Force Algorithm

- The brute force algorithm involves traversing the decision tree that choosing a path requires.
- A grid as shown above can be represented as a 2D list in python and using a backtracking recursive solution each path can be stored in the list.
- Since there is no constraints on the number of shortcuts (besides it being less than or equal to **K**), we make two backtracking calls at each instance: one to account for the fact that we chose a normal step to a cell which is consecutive and greater than the current and the other where we take a shortcut step as described above. Notice that there is only one shortcut step (at most, sometimes there is 0) from a particular cell in our path.
- We terminate each call when we reach the center room or the length of our path exceeds **K** (would be futile to continue down those paths).

Although we aren't computing all the possible paths, this method is rather computationally inefficient.

Problem Analysis

- The minimum number of steps required to reach the N^2 is *N - 1*. We can immediately reject anything smaller and accept K = *N - 1*.

- Another immediate decision-making heuristic is that we can reject any *K* value that is odd. This observation was deduced by taking some examples. Because of the symmetry involved in this problem, deductions can be made for easily.

- Let's define the concept of layers. For N = 5, we have that 1- 15 is the “first layer”, 16 - 23 is the “second layer”, and 24-25 is the “third layer”. An important observation is that once we reach a greater layer, we cannot move to a smaller layer number.

- When we take a shortcut, we are guaranteed to move into another layer of the grid with the exception of the corner cells of each layer (all the corner cells don’t even have shortcut options).

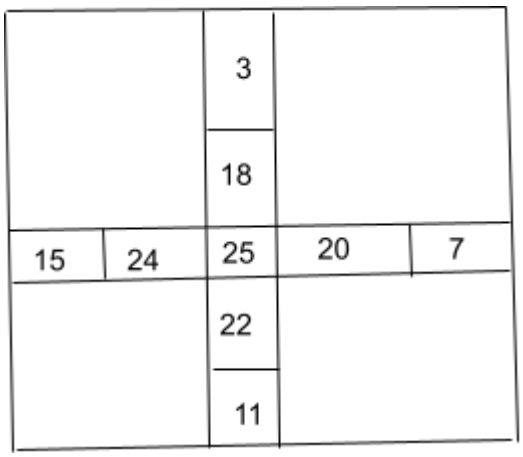
- We can easily determine the minimum number of steps to reach the cell and the maximum number of steps needed to reach the cell.

- For the first layer, there is only one way to get to a particular cell. This path has to be shortcut-free, which implies that it takes cell number - 1 step to reach there.

- Initially, we can consider the remaining number of steps to be $N^2 - 1 - k$ which is the number of short-cut free steps we want to take.

- In the code, the idea of a layer can be slightly modified to make it simpler. For N = 5, we have that the outermost layer is 1 - 16, then 17 - 24, then 25.

- Next, it is important to understand that for every row/column in a “layer” any arbitrary path from an arbitrary cell can be reduced to paths from the center cell. Hence, as we consider moving from one layer to another using shortcuts, we have to try moving from 4 center cells. For N = 5, the highlighted cells below for the center cells:



- In summary, have to traverse through the layers (number of layers - 1) and for each layer consider 4 cell entry points. For larger k, we want to take longer normal, shortcut free paths before opting for the shortcut and so we need to find a conditional involving the number of steps we are skipping.

Code Solution

```
def solve_spiral(N, K):
   %simple checks
   if K < N-1 or (K % 2) == 1:
       return "IMPOSSIBLE"
   res = []
   %the remain variable is represents the number of shortcut-free steps we want to
take
   remain = (N**2 - 1) - K
   num_layers = (N//2)
%reverse the numbering for easier calculation
   for l in range(num_layers, 0, -1):
       %the top left corner of every layer
       top_corner = (N**2) - (2*l + 1)**2 + 1
       for cell in range(4):
           jump = (8 * l - 1) - 2*cell
           jump_steps = jump - 1
           %only take a shortcut when we can keep remain >= 0
           if jump_steps > remain:
               continue
           print("took shortcut")
           remain -= jump_steps

           %the center cell for each layer is cell_number_start and the shortcut cell
destination is cell_number_end

           cell_number_start = (top_corner + l) + (2 * l * cell)
           cell_number_end = cell_number_start + jump
           res.append([cell_number_start, cell_number_end])
%once a shortcut is taken for that layer, we move to the next layer
           break
%break when we don't have sufficient steps (jump_steps >= 2)
       if remain <= 1:
           break
   if not remain:
       print(len(res))
       print()
```

```
        for [x,y] in res:
            print (str(x) + " " + str(y))
            print()
    else:
        print("IMPOSSIBLE")
```

Time Complexity : O (N)
Space Complexity: O (1)