Number Pattern Printing Problem

Problem Statement

Print a pattern of numbers from n to 1 in a square matrix of size $(2n-1) \times (2n-1)$ as shown below. Each number is separated by a single space.

For example, when $n = 4$, the output should be:

```
4 4 4 4 4 4 4
4 3 3 3 3 3 4
4 3 2 2 2 3 4
4 3 2 1 2 3 4
4 3 2 2 2 3 4
4 3 3 3 3 3 4
4 4 4 4 4 4 4
```

Input Format

The input contains a single integer n.

Constraints

- $1 \leq n \leq 1000$

Sample Input and Output

Input:

```
2
```

Output:

```
2 2 2
2 1 2
2 2 2
```

Input:

```
5
```

Output:

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

```
5 4 3 2 2 2 3 4 5
5 4 3 3 3 3 3 4 5
5 4 4 4 4 4 4 4 5
5 5 5 5 5 5 5 5 5
```

Solution Code in C

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n;
    scanf("%d", &n);

    int size = 2 * n - 1;  // total rows and columns

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            // Calculate minimum distance from edges (top, bottom, left, right)
            int minDist = i;
            if (j < minDist) minDist = j;
            if (size - 1 - i < minDist) minDist = size - 1 - i;
            if (size - 1 - j < minDist) minDist = size - 1 - j;

            // Print number based on minimum distance from edge
            printf("%d ", n - minDist);
        }
        printf("\n");
    }

    return 0;
}
```

Listing 1: Print concentric number pattern

Explanation

- The size of the output matrix is $(2n - 1)$ rows and columns.
- For each cell at position (i, j), find the minimum distance to the edges of the matrix:

$$\text{minDist} = \min(i, j, \text{size} - 1 - i, \text{size} - 1 - j)$$

- The number to print is then:

$$n - \text{minDist}$$

- This creates concentric layers of numbers from n at the outermost layer to 1 in the center.

Common Related Problems

- Printing concentric square patterns using characters or alphabets.
- Printing diamond or circular number patterns.

- Generating spiral matrices with numbers incrementing or decrementing.
- Matrix layer rotation problems.
- Pattern printing with palindromic rows or columns.

How to Make This Problem Harder

- **Different Shapes:** Extend to diamond or hexagonal shapes instead of squares.
- **Larger Inputs:** Optimize for very large n where naive printing may not be efficient.
- **Variable Step Decrease:** Instead of decreasing by 1 per layer, use a custom sequence or function.
- **Non-symmetric Patterns:** Generate similar patterns but with asymmetric edges or weighted layers.
- **3D Patterns:** Extend the concept to 3D arrays or cubes showing concentric shells.
- **Interactive Patterns:** Allow user-defined functions to decide the number based on position.