30 Pattern Programs

1. Right-Angled Triangle

This pattern forms a right-angled triangle with stars (*).

```
*
* *
* * *
* * * *
* * * * *
```

JAVA

```
public class RightAngleTriangle {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Python

```
n = 5  # Number of rows
for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()
```

2. Inverted Right-Angled Triangle

This pattern forms an inverted right-angled triangle with stars (*).

```
* * * * *
* * * *
* * *
* *
*
```

Java

```java
public class InvertedTriangle {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
n = 5  # Number of rows
for i in range(n, 0, -1):
    for j in range(i):
        print("*", end=" ")
    print()
```

3. Pyramid Pattern

This pattern forms a pyramid with stars (*).

```
    *
   ***
  *****
 *******
*********
```

Java

```
public class PyramidPattern {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = i; j < n; j++) {
                System.out.print(" "); // Print spaces
            }
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print("*"); // Print stars
            }
            System.out.println();
        }
    }
}
```

Python

```
n = 5  # Number of rows

for i in range(1, n + 1):

    print(" " * (n - i) + "*" * (2 * i - 1))
```

4. Diamond Pattern

This pattern forms a diamond shape with stars (*).

```
    *

   ***

  *****

 *******

*********

 *******

  *****

   ***

    *
```

Java

```
public class DiamondPattern {
  public static void main(String[] args) {
    int n = 5; // Half number of rows
    // Upper part of the diamond
    for (int i = 1; i <= n; i++) {
      for (int j = i; j < n; j++) {
        System.out.print(" "); // Print spaces
      }
      for (int k = 1; k <= (2 * i - 1); k++) {
        System.out.print("*"); // Print stars
      }
      System.out.println();
    }
    // Lower part of the diamond
    for (int i = n - 1; i >= 1; i--) {
      for (int j = n; j > i; j--) {
        System.out.print(" "); // Print spaces
      }
      for (int k = 1; k <= (2 * i - 1); k++) {
        System.out.print("*"); // Print stars
      }
      System.out.println();
    }
  }
}
```

Python

```
n = 5  # Half number of rows
# Upper part of the diamond
for i in range(1, n + 1):
    print(" " * (n - i) + "*" * (2 * i - 1))
# Lower part of the diamond
for i in range(n - 1, 0, -1):
    print(" " * (n - i) + "*" * (2 * i - 1))
```

5. Number Pyramid

This pattern prints numbers instead of stars, forming a pyramid.

```
    1
   222
  33333
 4444444
555555555
```

Java

```
public class NumberPyramid {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = i; j < n; j++) {
                System.out.print(" "); // Print spaces
            }
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print(i); // Print numbers
            }
            System.out.println();
        }
    }
}
```

Python

```
n = 5  # Number of rows

for i in range(1, n + 1):

    print(" " * (n - i) + str(i) * (2 * i - 1))
```

6. Right-Angled Triangle (with numbers)

This pattern uses numbers rather than stars.

```
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5
```

Java

```java
public class NumberTriangle {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
n = 5  # Number of rows

for i in range(1, n + 1):

    for j in range(1, i + 1):

        print(j, end=" ")

    print()
```

7 Inverted Number Triangle

This pattern starts with a larger set of numbers and decreases with each row.

```
1 2 3 4 5

1 2 3 4

1 2 3

1 2

1
```

Java

```java
public class InvertedNumberTriangle {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
n = 5  # Number of rows

for i in range(n, 0, -1):

    for j in range(1, i + 1):

        print(j, end=" ")

    print()
```

8 Hollow Diamond

A more complex pattern where stars form a hollow diamond shape.

```
    *
   * *
  *   *
 *     *
*********
 *     *
  *   *
   * *
    *
```

Java

```
public class HollowDiamond {
    public static void main(String[] args) {
        int n = 5; // Half number of rows
        // Upper part of the hollow diamond
        for (int i = 1; i <= n; i++) {
            for (int j = i; j < n; j++) {
                System.out.print(" "); // Print spaces
            }
            for (int k = 1; k <= (2 * i - 1); k++) {
                if (k == 1 || k == (2 * i - 1)) {
                    System.out.print("*"); // Print stars
                } else {
                    System.out.print(" "); // Hollow space
                }
            }
            System.out.println();
        }
        // Lower part of the hollow diamond
        for (int i = n - 1; i >= 1; i--) {
            for (int j = n; j > i; j--) {
                System.out.print(" "); // Print spaces
            }
            for (int k = 1; k <= (2 * i - 1); k++) {
                if (k == 1 || k == (2 * i - 1)) {
                    System.out.print("*"); // Print stars
                } else {
                    System.out.print(" "); // Hollow space
            }
          }
          System.out.println();
        }
    }
}
```

Python

```
n = 5  # Half number of rows
# Upper part of the hollow diamond
for i in range(1, n + 1):
    print(" " * (n - i), end="")
    for j in range(1, 2 * i):
        if j == 1 or j == (2 * i - 1):
            print("*", end="")
        else:
            print(" ", end="")
    print()
# Lower part of the hollow diamond
for i in range(n - 1, 0, -1):
    print(" " * (n - i), end="")
    for j in range(1, 2 * i):
        if j == 1 or j == (2 * i - 1):
            print("*", end="")
        else:
            print(" ", end="")
    print()
```

9 Floyd's Triangle (with Numbers)

This pattern is a number-based triangle where each row contains sequential numbers.

```
1

2 3

4 5 6

7 8 9 10

11 12 13 14 15
```

Java

```java
public class FloydsTriangle {
    public static void main(String[] args) {
        int n = 5; // Number of rows
        int num = 1; // Start with number 1
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(num + " ");
                num++;
            }
            System.out.println();
        }
    }
}
```

Python

```python
n = 5  # Number of rows
num = 1  # Start with number 1
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(num, end=" ")
        num += 1
    print()
```

10 Butterfly Pattern

A butterfly-shaped pattern made with stars.

```
*        *

**      **

***    ***

****  ****

**********

****  ****

***    ***

**      **

*        *
```

Java

```
public class ButterflyPattern {
public static void main(String[] args) {
        int n = 5; // Number of rows
        // Upper part of the butterfly
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            for (int j = 1; j <= 2 * (n - i); j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        // Lower part of the butterfly
        for (int i = n - 1; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            for (int j = 1; j <= 2 * (n - i); j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Python

```
n = 5  # Number of rows
# Upper part of the butterfly
for i in range(1, n + 1):
    print("*" * i + " " * (2 * (n - i)) + "*" * i)
# Lower part of the butterfly
for i in range(n - 1, 0, -1):
    print("*" * i + " " * (2 * (n - i)) + "*" * i)
```

11 Two Diagonals Crossing Each Other (Stars)

```
*       *
  *   *
    *
  *   *
*       *
```

Java

```
public class DiagonalCrossPattern {
    public static void main(String[] args) {
        int n = 5; // Size of the square matrix
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                // Conditions for the two diagonals
                if (i == j || i + j == n - 1) {
                    System.out.print("* ");  // Star at diagonal
                } else {
                    System.out.print("  ");  // Space elsewhere
                }
            }
            System.out.println();
        }
    }
}
```

Python

```
n = 5  # Size of the square matrix
for i in range(n):
    for j in range(n):
        # Conditions for the two diagonals
        if i == j or i + j == n - 1:
            print("*", end=" ")  # Star at diagonal
        else:
            print(" ", end=" ")  # Space elsewhere
    print()
```

12 Square Pattern with 5 Lines and 5 Stars in Each Line

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
public class SquarePattern {
    public static void main(String[] args) {
        int n = 5; // Number of rows and columns (5x5 grid)
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print("* ");  // Print star in each column
            }
            System.out.println();  // Move to the next line
        }
    }
}
```

```
n = 5  # Number of rows and columns (5x5 grid)
for i in range(n):
    for j in range(n):
        print("*", end=" ")  # Print star in each column
    print()  # Move to the next line
```

13 Square Pattern with Border and Empty Center

```
* * * * *
*       *
*       *
*       *
* * * * *
```

Java

```
public class HollowSquarePattern {
    public static void main(String[] args) {
        int n = 5;                    // Size of the square (5x5 grid)
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                // Print stars on the border, otherwise print space
                if (i == 0 || i == n - 1 || j == 0 || j == n - 1) {
                    System.out.print("* ");
                } else {
                    System.out.print("  ");
                }
            }
            System.out.println();           // Move to the next line
        }
    }
}
```

Python

```
n = 5                              # Size of the square (5x5 grid)
for i in range(n):
    for j in range(n):
        # Print stars on the border, otherwise print space
        if i == 0 or i == n - 1 or j == 0 or j == n - 1:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()                            # Move to the next line
```

14. You are given a pattern that alternates between rows of "X"s and rows of numbers. The numbers start at 9 and decrease by 2 with each subsequent occurrence. Your task is to print the given pattern using a programming language of your choice. The pattern looks like this:

```
XXXXXXXXXX
9999999999
XXXXXXXXXX
7777777777
XXXXXXXXXX
6666666666
```

Java

```
public class PatternPrinter {
  public static void main(String[] args) {
    int[] numbers = {9, 7, 6}; // The numbers to print
    int numLines = numbers.length; // How many lines with numbers
    int length = 10; // Length of the "X" string and each number string

    // Loop through the pattern
    for (int i = 0; i < numLines; i++) {
      if (i % 2 == 0) {
        // Print "X" for even index rows (0, 2, 4, ...)
        for (int j = 0; j < length; j++) {
          System.out.print("X");
        }
      } else {
        // Print number for odd index rows (1, 3, 5, ...)
        for (int j = 0; j < length; j++) {
          System.out.print(numbers[i / 2]);
        }
      }
      System.out.println(); // Move to the next line after each row
    }
  }
}
```

Python

```
def print_pattern():
    numbers = [9, 7, 6]  # The numbers to print
    length = 10  # Length of each "X" string and number string

    for i in range(len(numbers) + 1):  # Looping through the pattern
        if i % 2 == 0:
            # Print "X" for even index rows (0, 2, 4, ...)
            print("X" * length)
        else:
            # Print the number for odd index rows (1, 3, 5, ...)
            print(str(numbers[i // 2]) * length)

# Call the function to print the pattern
print_pattern()
```

15 .Your program must be dynamic It should work for any odd numbers (excluding 1 and any negative numbers). It should not run for any even numbers. for the second line it should be 3 gap.. 3rd line 1

```
*******
**   **  n=7
*** ***
*******
*** ***
**   **
*******
```

```
*********
**     **
***   ***        n= 9
*********
***   ***
**     **
*********
```

Java

```
import java.util.Scanner;

public class DynamicPatternPrinter {

  public static void main(String[] args) {
    // Input Scanner for the number of columns
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter an odd number greater than 1: ");
    int n = scanner.nextInt();

    // Check if the number is valid (odd and greater than 1)
    if (n <= 1 || n % 2 == 0) {
      System.out.println("Invalid input! The number should be an odd number greater than 1.");
      return;
    }

    // Loop through the rows
    for (int i = 0; i < n; i++) {
      if (i == 0 || i == n - 1) {
        // Print the first and last row with all stars
        for (int j = 0; j < n; j++) {
            System.out.print("*");
        }
      }
      else if (i == 1 || i == n - 2) {
        // Print the second and second-to-last row: **...** with (n-4) spaces
        System.out.print("**");
        for (int j = 0; j < n - 4; j++) {
            System.out.print(" ");
        }
        System.out.print("**");
      }
      else if (i == 2 || i == n - 3) {
        // Print the third and fifth row: ***...*** with (n-6) spaces
        System.out.print("***");
        for (int j = 0; j < n - 6; j++) {
            System.out.print(" ");
        }
        System.out.print("***");
      }
      else if (i == n / 2) {
        // Print the middle row: all stars
        for (int j = 0; j < n; j++) {
            System.out.print("*");
        }
      }

      // Move to the next line after each row
      System.out.println();
    }

    scanner.close();  // Close the scanner
  }
}
```

Python

```
def print_pattern():
    # Input for the number of columns
    n = int(input("Enter an odd number greater than 1: "))

    # Check if the number is valid (odd and greater than 1)
    if n <= 1 or n % 2 == 0:
        print("Invalid input! The number should be an odd number greater than 1.")
        return

    # Loop through the rows
    for i in range(n):
        if i == 0 or i == n - 1:
            # Print the first and last row with all stars
            print("*" * n)
        elif i == 1 or i == n - 2:
            # Print the second and second-to-last row: **...** with (n-4) spaces
            print("**" + " " * (n - 4) + "**")
        elif i == 2 or i == n - 3:
            # Print the third and fifth row: ***...*** with (n-6) spaces
            print("***" + " " * (n - 6) + "***")
        elif i == n // 2:
            # Print the middle row: all stars
            print("*" * n)

# Call the function to print the pattern
print_pattern()
```

16. Number Pyramid

```
1
121
12321
1234321
123454321
```

Java

```java
public class NumberPyramid {
    public static void main(String[] args) {
        int n = 5;  // Number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j);
            }
            for (int j = i - 1; j >= 1; j--) {
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

Python

```python
def number_pyramid(n):
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(j, end="")
        for j in range(i - 1, 0, -1):
            print(j, end="")
        print()

number_pyramid(5)
```

17. Hollow Inverted Right Angle Triangle

```
*****
*  *
* *
**
*
```

Java

```java
public class HollowInvertedRightAngle {
    public static void main(String[] args) {
        int n = 5;
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                if (j == 1 || j == i || i == n) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

Python

```python
def hollow_inverted_right_angle(n):
    for i in range(n, 0, -1):
        for j in range(1, i + 1):
            if j == 1 or j == i or i == n:
                print("*", end="")
            else:
                print(" ", end="")
        print()

hollow_inverted_right_angle(5)
```

18. Number Spiral

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Java

```java
public class NumberSpiral {
    public static void main(String[] args) {
        int n = 4;
        int[][] matrix = new int[n][n];
        int num = 1;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = num++;
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
def number_spiral(n):
    matrix = [[0] * n for _ in range(n)]
    num = 1
    for i in range(n):
        for j in range(n):
            matrix[i][j] = num
            num += 1

    for row in matrix:
        print(" ".join(map(str, row)))

number_spiral(4)
```

19. Pascal’s Triangle

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

Java

```java
public class PascalsTriangle {
    public static void main(String[] args) {
        int n = 5;
        int[][] triangle = new int[n][n];

        for (int i = 0; i < n; i++) {
            triangle[i][0] = triangle[i][i] = 1;
            for (int j = 1; j < i; j++) {
                triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.print(triangle[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
def pascals_triangle(n):
    triangle = [[0] * n for _ in range(n)]
    for i in range(n):
        triangle[i][0] = triangle[i][i] = 1
        for j in range(1, i):
            triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j]

    for i in range(n):
        print(" ".join(map(str, triangle[i][:i+1])))

pascals_triangle(5)
```

20. Hourglass with Stars

```
*******
 *****
  ***
   *
  ***
 *****
*******
```

Java

```
public class HourglassWithStars {
  public static void main(String[] args) {
    int n = 7;
    for (int i = 0; i < n / 2; i++) {
      for (int j = 0; j < i; j++) {
        System.out.print(" ");
      }
      for (int j = 0; j < n - 2 * i; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
    for (int i = n / 2; i < n; i++) {
      for (int j = 0; j < n - i - 1; j++) {
        System.out.print(" ");
      }
      for (int j = 0; j < 2 * i - n + 1; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

Python

```
def hourglass_with_stars(n):
  for i in range(n // 2):
    print(" " * i + "*" * (n - 2 * i))
  for i in range(n // 2, n):
    print(" " * (n - i - 1) + "*" * (2 * i - n + 1))

hourglass_with_stars(7)
```

21. Zigzag Number Pattern

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

Java

```java
public class ZigzagNumberPattern {
    public static void main(String[] args) {
        int n = 5;
        int num = 1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                System.out.print(" ");
            }
            for (int j = i; j < n; j++) {
                System.out.print(num++ + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```python
def zigzag_number_pattern(n):
    num = 1
    for i in range(n):
        print(" " * i + " ".join(str(num + j) for j in range(n - i)))
        num += (n - i)

zigzag_number_pattern(5)
```

22.Hollow Number Square

```
1  2   3  4
5          6
7          8
9 10 11 12
```

Java

```java
public class HollowNumberSquare {
   public static void main(String[] args) {
      int n = 4;  // Size of the square
      int num = 1;
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
            if (i == 0 || i == n - 1 || j == 0 || j == n - 1) {
               System.out.print(num++ + " ");
            } else {
               System.out.print("  ");
            }
         }
         System.out.println();
      }
   }
}
```

Python

```python
def hollow_number_square(n):
   num = 1
   for i in range(n):
      for j in range(n):
         if i == 0 or i == n - 1 or j == 0 or j == n - 1:
            print(num, end=" ")
            num += 1
         else:
            print(" ", end=" ")
      print()

hollow_number_square(4)
```

23 Hollow Diamond

```
    1
   121
 1      1
121   121
  1      1
    121
      1
```

Java

```
public class HollowDiamond {
    public static void main(String[] args) {
        int n = 5;  // Middle row of the diamond
        // Upper half of the diamond
        for (int i = 1; i <= n; i++) {
            for (int j = i; j < n; j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= i; j++) {
                if (j == 1 || j == i) {
                    System.out.print(j);
                } else {
                    System.out.print(" ");
                }
            }
            for (int j = i - 1; j >= 1; j--) {
                if (j == 1 || j == i) {
                    System.out.print(j);
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
        // Lower half of the diamond
        for (int i = n - 1; i >= 1; i--) {
            for (int j = i; j < n; j++) {
                System.out.print(" ");
            }
            for (int j = 1; j <= i; j++) {
                if (j == 1 || j == i) {
                    System.out.print(j);
                } else {
                    System.out.print(" ");
                }
            }
            for (int j = i - 1; j >= 1; j--) {
                if (j == 1 || j == i) {
                    System.out.print(j);
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

Python

```
def hollow_diamond(n):
    # Upper half of the diamond
    for i in range(1, n + 1):
        print(" " * (n - i), end="")
        for j in range(1, i + 1):
            if j == 1 or j == i:
                print(j, end="")
            else:
                print(" ", end="")
        for j in range(i - 1, 0, -1):
            if j == 1 or j == i:
                print(j, end="")
            else:
                print(" ", end="")
        print()

    # Lower half of the diamond
    for i in range(n - 1, 0, -1):
        print(" " * (n - i), end="")
        for j in range(1, i + 1):
            if j == 1 or j == i:
                print(j, end="")
            else:
                print(" ", end="")
        for j in range(i - 1, 0, -1):
            if j == 1 or j == i:
                print(j, end="")
            else:
                print(" ", end="")
    print()

hollow_diamond(5)
```

24 Spiral of Numbers (Counterclockwise)

```
1 2 3
8 9 4
7 6 5
```

Java

```java
public class NumberSpiralCounterclockwise {
    public static void main(String[] args) {
        int n = 3;  // Size of the matrix (should be odd)
        int[][] spiral = new int[n][n];
        int num = 1;
        int top = 0, bottom = n - 1, left = 0, right = n - 1;

        while (top <= bottom && left <= right) {
            // Fill the left column
            for (int i = top; i <= bottom; i++) {
                spiral[i][left] = num++;
            }
            left++;

            // Fill the bottom row
            for (int i = left; i <= right; i++) {
                spiral[bottom][i] = num++;
            }
            bottom--;

            // Fill the right column
            for (int i = bottom; i >= top; i--) {
                spiral[i][right] = num++;
            }
            right--;

            // Fill the top row
            for (int i = right; i >= left; i--) {
                spiral[top][i] = num++;
            }
            top++;
        }

        // Print the spiral matrix
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(spiral[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Python

```
def number_spiral_counterclockwise(n):
    spiral = [[0] * n for _ in range(n)]
    num = 1
    top, bottom, left, right = 0, n - 1, 0, n - 1

    while top <= bottom and left <= right:
        # Fill the left column
        for i in range(top, bottom + 1):
            spiral[i][left] = num
            num += 1
        left += 1

        # Fill the bottom row
        for i in range(left, right + 1):
            spiral[bottom][i] = num
            num += 1
        bottom -= 1

        # Fill the right column
        for i in range(bottom, top - 1, -1):
            spiral[i][right] = num
            num += 1
        right -= 1

        # Fill the top row
        for i in range(right, left - 1, -1):
            spiral[top][i] = num
            num += 1
        top += 1

    # Print the spiral
    for row in spiral:
        print(" ".join(map(str, row)))

number_spiral_counterclockwise(3)
```

25 Alternating Odd and Even Numbers in a Square

Java

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

```java
public class AlternatingOddEvenSquare {
    public static void main(String[] args) {
        int n = 4;  // Size of the square
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if ((i + j) % 2 == 0) {
                    System.out.print(1 + " ");
                } else {
                    System.out.print(2 + " ");
                }
            }
            System.out.println();
        }
    }
}
```

Python

```python
def alternating_odd_even_square(n):
    for i in range(n):
        for j in range(n):
            if (i + j) % 2 == 0:
                print(1, end=" ")
            else:
                print(2, end=" ")
        print()

alternating_odd_even_square(4)
```