Subject: Computer Programming: Python

Faculty: Premanand S / SENSE / VIT-CC

FAT Question: 3

Kithan, a mathematical genius, and Krishna, a Pythonista, decided to create a Python program to generate a Magic Square. A Magic Square is a square grid where the sum of the numbers in each row, column, and both diagonals are the same.

Kithan asked Krishna to implement the logic for creating a Magic Square in Python. Krishna was unfamiliar with the concept, so Kithan explained the steps. Below is the explanation of how the program should work:

Instructions:

1. **Input**: The program should prompt the user to enter a natural odd number (such as 3, 5, 7, 9, etc.).
2. **Square Size**: If the user enters an odd number (let's say n = 5), then a square grid with 5 rows and 5 columns should be created. The grid should contain the numbers 1 to n*n.
3. **Magic Square**: The goal is to arrange the numbers in the grid such that the sum of each row, column, and both diagonals is the same. This sum is referred to as the **Magic Constant** or **Magic Sum**. For example, if n = 3, the sum of each row, column, and diagonal should be 15.
4. **Condition**: Only odd numbers are allowed for n. Even numbers will not work to create a Magic Square.
5. **Example**: For n = 3, the **Magic Square** would look like this:

Sum of each row or column: 15

2 7 6

9 5 1

4 3 8

The sum of each row, column, and diagonal in this square is 15, which is the **Magic Sum**.

Output:

The program should output the Magic Square where the sum of each row, column, and diagonal is the same, and it should print the **Magic Sum** for the square.

Example 1:

For input n = 3, the Magic Square should look like this:

Sum of each row or column: 15

2 7 6

9 5 1

4 3 8

Example 2:

For input n = 5, the Magic Square should look like this:

Sum of each row or column: 65

17 24 1 8 15

23 5 7 14 16

4 6 13 20 22

10 12 19 21 3

11 18 25 2 9

The program should ensure that:

1. The number provided by the user is odd (e.g., 3, 5, 7, etc.).
2. The sum of all rows, columns, and diagonals equals the same **Magic Sum**. (Note: No libraries should be used)

Solution:

Input:

The program asks the user to input a **natural odd number (n)**, where n represents the size of the magic square. Example input: 3, 5, 7, etc.

Output:

1. The output is a **magic square** of size n x n, where the numbers from 1 to n*n are arranged in such a way that the sum of each row, column, and both diagonals is the same.
2. It also displays the **Magic Sum**, which is the sum of any row, column, or diagonal (they will all be the same).

Example output:

For n = 3:

Sum of each row or column: 15

2 7 6
9 5 1
4 3 8

For n = 5:

Sum of each row or column: 65

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

Processing:

1. I**nput Validation**: The program checks that the entered number is a valid odd number. If the number is not odd, the program will ask the user to input again.
2. **Magic Square Generation (Siamese Method)**:

- The program initializes an empty n x n grid.
- It starts by placing the number 1 in the middle of the top row.
- For each subsequent number, it attempts to place it diagonally **up-right**. If this move takes the program out of bounds of the grid, it wraps around.
- If the intended cell is already filled, the number is placed directly **below** the last placed number.

3. **Magic Sum Calculation**: The sum of any row, column, or diagonal is calculated to verify the magic property of the square. This sum is known as the **Magic Sum**.
4. **Output**: The program outputs the grid as the magic square and the Magic Sum.

Algorithm:

1. **Input**: Take an odd number n as input.
2. **Grid Initialization**: Create an empty n x n grid filled with zeros.
3. **First Step**: Place 1 in the middle of the top row (position (0, n//2)).
4. **Loop (for each number from 1 to n*n)**:

- Place the next number in the current cell.
- Calculate the next position (move diagonally up-right).
- If moving out of bounds, wrap around.
- If the new position is already occupied, move directly down from the last placed number.

5. **Magic Sum**: Calculate the sum of the first row, which should be equal to the Magic Sum. Verify this sum holds for all rows, columns, and diagonals.
6. **Output**: Print the magic square and the Magic Sum.

Solution Alternative:

1. **Brute Force/Exhaustive Search**: An alternative approach could involve generating all possible configurations of numbers and checking if the sum of each row, column, and diagonal is equal. However, this is inefficient for larger n because it would require checking factorial number of configurations.
2. **Using Libraries**: If allowed, libraries like numpy or sympy could simplify operations like matrix manipulation and checking sums.

However, the Siamese method is optimal and efficient for odd-sized magic squares, as it directly constructs the solution without the need for exhaustive search or additional computation.

Code:

```
def generate_magic_square(n):
    # Initialize an n x n grid with zeroes
    magic_square = [[0] * n for _ in range(n)]

    # Start position for 1 (middle of the top row)
    row, col = 0, n // 2

    # Fill the magic square
    for num in range(1, n*n + 1):
        magic_square[row][col] = num
        # Move to the next position: diagonally up-right
        new_row, new_col = (row - 1) % n, (col + 1) % n

        # If the calculated cell is already filled, move down
        if magic_square[new_row][new_col] != 0:
            row += 1
        else:
            row, col = new_row, new_col

    # Calculate the magic sum (sum of any row, column, or diagonal)
    magic_sum = sum(magic_square[0])

    return magic_square, magic_sum

def print_magic_square(magic_square, magic_sum):
    print(f"Sum of each row or column: {magic_sum}")
    for row in magic_square:
        print("  ".join(str(num) for num in row))

def main():
    while True:
        try:
            # Input for size of the magic square (must be odd)
```

```
            n = int(input("Enter an odd number for the magic square size: "))
            if n % 2 == 1:
                break
            else:
                print("Please enter an odd number.")
        except ValueError:
            print("Invalid input! Please enter a valid integer.")

    magic_square, magic_sum = generate_magic_square(n)
    print_magic_square(magic_square, magic_sum)

# Call the main function
main()
```

Output:

```
Enter an odd number for the magic square size: 5
Sum of each row or column: 65
17  24  1  8  15
23  5  7  14  16
4  6  13  20  22
10  12  19  21  3
11  18  25  2  9

Enter an odd number for the magic square size: 3
Sum of each row or column: 15
8  1  6
3  5  7
4  9  2
```