Recursive Collatz Count

Given an integer n, count the number of steps it takes to reach 1 using the Collatz conjecture with a **recursive approach**.

Collatz Conjecture: Starting with any positive integer n, repeatedly apply the following operation:

- If n is even, divide it by 2: $n \leftarrow n/2$
- If n is odd, multiply by 3 and add 1: $n \leftarrow 3n + 1$

The conjecture states that this sequence will always eventually reach 1, regardless of the starting value.

Important: Your solution **MUST** use recursion. Any non-recursive solution (using loops) will receive zero points from the judge, regardless of correctness.

Input Format

A single integer n

Output Format

A single integer representing the number of steps to reach 1.

Constraints

- $1 \leq n \leq 100000$ (to avoid stack overflow with recursion)
- Each test case will have multiple numbers to process

Sample Input

```
10 6 3
```

Sample Output

```
6 8 7
```

Explanation

- $n = 10$: $10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$ (6 steps)
- $n = 6$: $6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$ (8 steps)
- $n = 3$: $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$ (7 steps)

Implementation

Goal: Fill in the following function using **recursion only**:

```
def recur_collatz_count(n: int) -> int:
    ...
    return ...

if __name__ == "__main__":
    # You can test anything inside this block and can send it to grader
    # The grader will use only the function that you have implemented
    # !!! DO NOT write anything outside this block
    print(recur_collatz_count(10))  # Should print 6
    print(recur_collatz_count(6))   # Should print 8
    print(recur_collatz_count(3))   # Should print 7
```

WARNING: The judge will automatically check if your solution uses recursion. Any non-recursive solution (using loops like `while`, `for`, etc.) will receive **zero points** regardless of correctness.