Calculate the Sum of Digits of a number using recursion

The provided C++ program calculates the sum of digits in a given number using a recursive approach.

Recursive function to calculate the sum of digits in a number

1. Inside the **addDigit** function, there are two cases:
   - Base Case: If the **num** is less than or equal to 0, it means the number is non-positive, and there are no digits to add. In this case, the function returns 0.
   - Recursive Case: If the **num** is positive, the function calculates the sum of digits by adding the last digit (obtained using **num % 10**) to the sum of the remaining digits (obtained using **num / 10**) recursively.

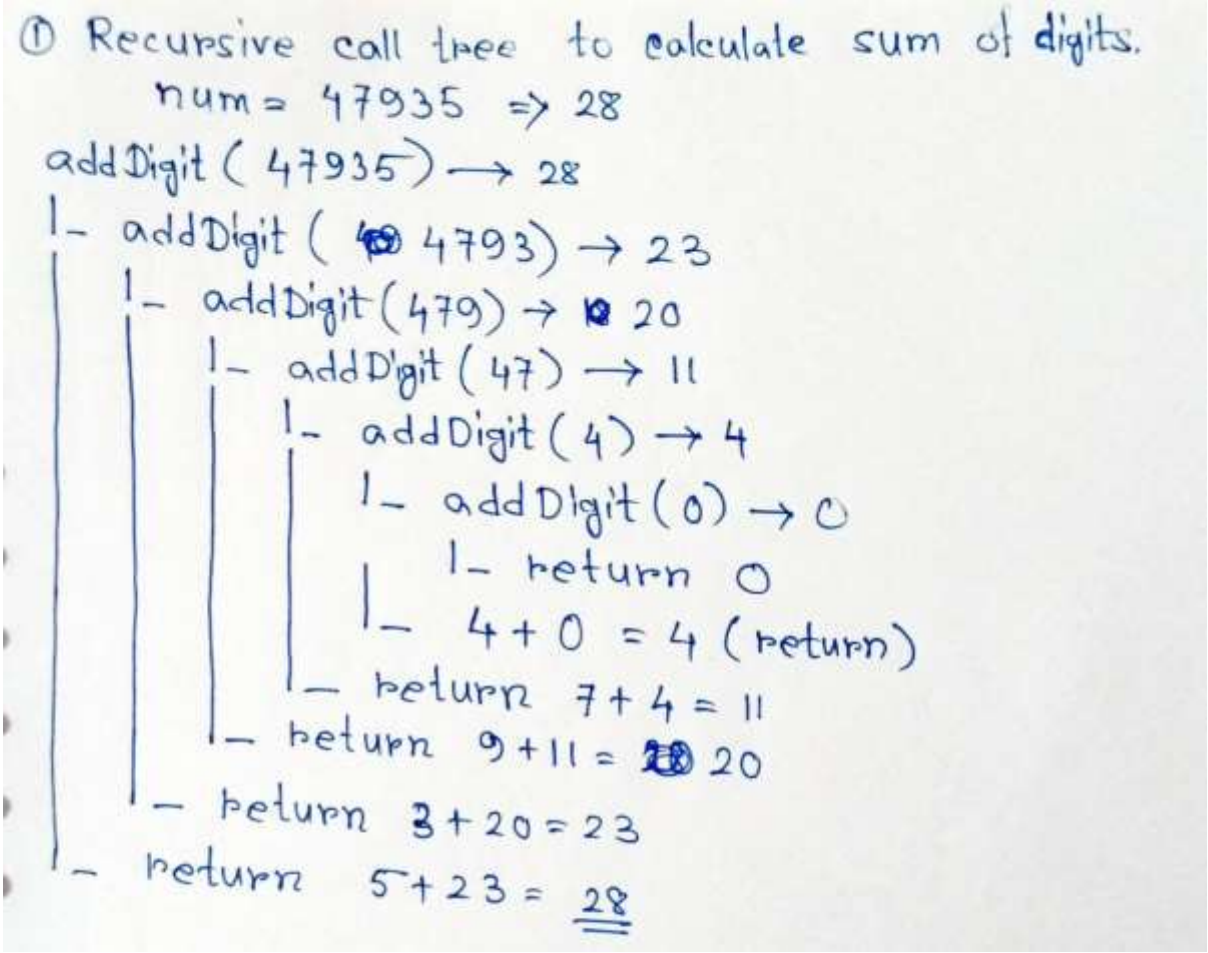
Time Complexity:

Time complexity of the addDigit function is O(d), where d is the number of digits in the input number.

Space Complexity:

The space complexity of the program is O(d), where d is the number of digits in the input number. This is because the recursive calls in the addDigit function create new frames on the call stack, and in the worst case, there can be d recursive calls, leading to O(d) space consumption on the call stack.

Recursive call stack of the approach:



① Recursive call tree to calculate sum of digits.

```
      num = 47935  => 28
 addDigit ( 47935 ) ⟶ 28
 |- addDigit ( 4793 ) → 23
 |   |- addDigit (479) → 20
 |   |   |- addDigit ( 47 ) ⟶ 11
 |   |   |   |- addDigit ( 4 ) → 4
 |   |   |   |   |- addDigit ( 0 ) → 0
 |   |   |   |   |     |- return 0
 |   |   |   |   |- 4 + 0 = 4 (return)
 |   |   |   |- return 7 + 4 = 11
 |   |   |- return 9 + 11 = 20
 |   |- return 3 + 20 = 23
 |- return 5 + 23 = 28
```