Day 19: Find the Missing Number

Gati Goyal

"Good code is its own best documentation."
— Steve McConnell

1 Introduction

Finding the missing number in an array is a classic problem that can be solved using mathematical formulas or bitwise operations. The array contains $n - 1$ integers ranging from 1 to n, with exactly one number missing.

2 Problem Statement

Problem: Find the missing number in an array of size $n - 1$ containing numbers from 1 to n. **Hint:** Use the formula for the sum of the first n natural numbers:

$$\text{Sum} = \frac{n \times (n+1)}{2}.$$

Edge Case: Handle arrays with no missing numbers or duplicate entries.

3 Algorithm

1. Calculate the expected sum of the first n natural numbers using the formula:

$$\text{Sum} = \frac{n \times (n+1)}{2}.$$

2. Calculate the actual sum of the elements in the array.

3. The missing number is the difference between the expected sum and the actual sum.

4 Code

```c
#include <stdio.h>

int findMissingNumber(int arr[], int n) {
    int expectedSum = n * (n + 1) / 2;
    int actualSum = 0;

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

    return expectedSum - actualSum;
}

int main() {
    int n;

    printf("Enter the value of n (size of the full array): ");
    scanf("%d", &n);

    int arr[n - 1];
    printf("Enter the elements of the array: ");
    for (int i = 0; i < n - 1; i++) {
        scanf("%d", &arr[i]);
    }

    int missingNumber = findMissingNumber(arr, n);
    printf("The missing number is: %d\n", missingNumber);

    return 0;
}
```

5 Alternate Approach: XOR Method

The XOR method is another efficient way to find the missing number:

- XOR all the numbers from 1 to n.
- XOR all the elements in the array.
- XOR of the two results gives the missing number.

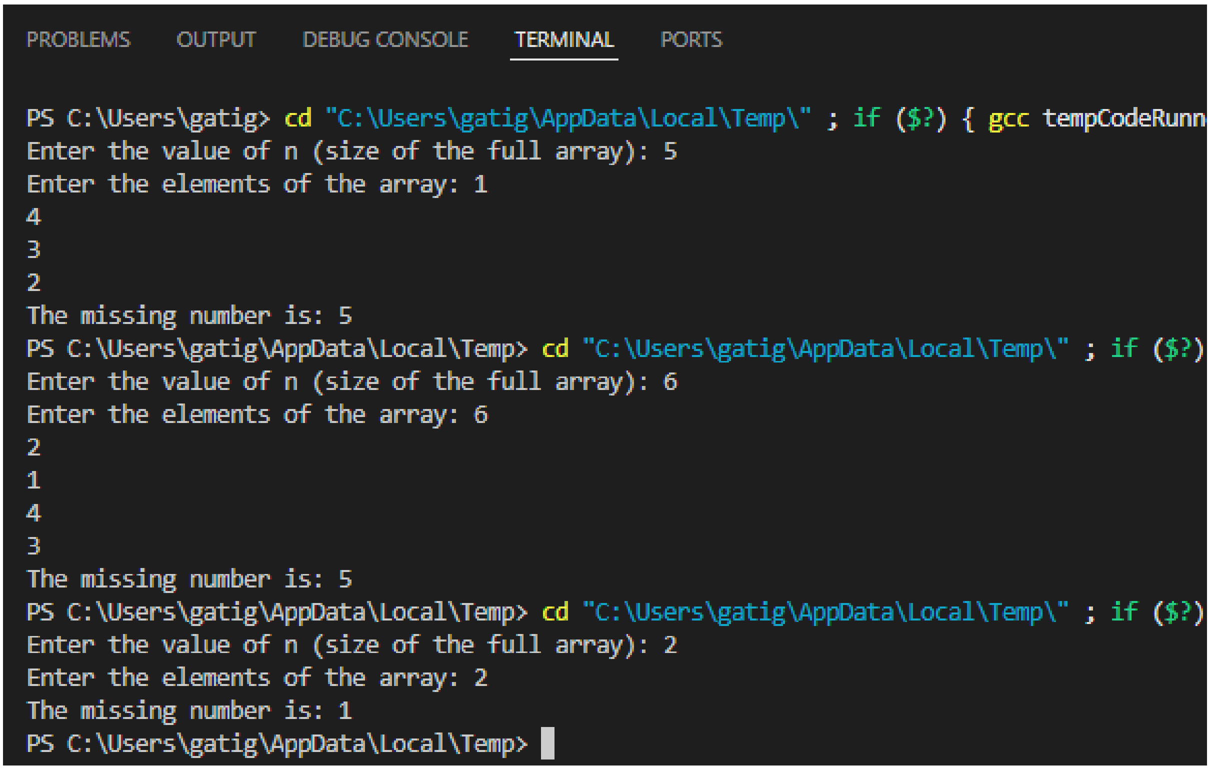
6 Complexity Analysis

- **Time Complexity:** $O(n)$ (single traversal of the array).
- **Space Complexity:** $O(1)$ (no additional memory required).

7 Examples and Edge Cases

Input Array	Missing Number	Explanation
{1, 2, 4, 5, 6}	3	Sum = 21, Actual Sum = 18, Missing = 3
{2, 3, 1, 5}	4	Sum = 15, Actual Sum = 11, Missing = 4
{1, 2, 3, 4, 5}	6	Expected case with $n = 6$

8 Output



```
PS C:\Users\gatig> cd "C:\Users\gatig\AppData\Local\Temp\" ; if ($?) { gcc tempCodeRunn
Enter the value of n (size of the full array): 5
Enter the elements of the array: 1
4
3
2
The missing number is: 5
PS C:\Users\gatig\AppData\Local\Temp> cd "C:\Users\gatig\AppData\Local\Temp\" ; if ($?)
Enter the value of n (size of the full array): 6
Enter the elements of the array: 6
2
1
4
3
The missing number is: 5
PS C:\Users\gatig\AppData\Local\Temp> cd "C:\Users\gatig\AppData\Local\Temp\" ; if ($?)
Enter the value of n (size of the full array): 2
Enter the elements of the array: 2
The missing number is: 1
PS C:\Users\gatig\AppData\Local\Temp>
```

Figure 1: Program Output Screenshot

9 Conclusion

The problem of finding the missing number demonstrates the efficiency of mathematical formulas and bitwise operations in problem-solving. The formula-based method is intuitive, while the XOR approach is computationally elegant, making both valuable tools for solving similar problems.