Practical - 1

Aim:

Implement a function for each of following problems and count the number of steps executed/Time taken by each function on various inputs and write complexity of each function. Also draw a comparative chart. In each of the following function N will be passed by user.

1. To calculate sum of 1 to N number using loop.
2. To calculate sum of 1 to N number using equation.
3. To calculate sum of 1 to N numbers using recursion.

Code:

```
#include <stdio.h>

int count_ite = 0;
int count_rec = 0;
int count_eq = 0;

int sum_i_142(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        count_ite++;
        sum += i;
    }
    return sum;
}

int sum_e_142(int n) {
    int sum = 1;
    sum = (n * (n + 1)) / 2;
    count_eq++;
    return sum;
}
```

```
int sum_r_142(int n) {
    if (n == 0) {
        count_rec++;
        return 0;
    }
    else
    if (n == 1) {
        count_rec++;
        return 1;
    }
    else {
        count_rec++;
        return n + sum_r_142(n - 1);
    }
}

int main() {
    int n;
    printf("Give Me Number To Sum of Numbers from 1 to N You want Sum of: ");
    scanf("%d", &n);
    int result_ite = sum_i_142(n);
    printf("\n-----------------------------------------");
    printf("\nEnroll Number: 142");
    printf("\nLoop Method");
    printf("\nSum of 1 to %d Number is: %d", n, result_ite);
    printf("\nCount of Step of Algorithm is : %d", count_ite);
    printf("\n-----------------------------------------");
    int result_eq = sum_e_142(n);
    printf("\n-----------------------------------------");
    printf("\nEnroll Number: 142");
    printf("\nEquation Method");
    printf("\nSum of 1 to %d Number is: %d", n, result_eq);
    printf("\nCount of Step of Algorithm is : %d", count_eq);
    printf("\n-----------------------------------------");
```

```
    int result_rec = sum_r_142(n);
    printf("\n----------------------------------------");
    printf("\nEnroll Number: 142");
    printf("\nRecursive Method");
    printf("\nSum of 1 to %d Number is: %d", n, result_rec);
    printf("\nCount of Step of Algorithm is : %d", count_rec);
    printf("\n----------------------------------------");
    return 0;
}
```

Output:

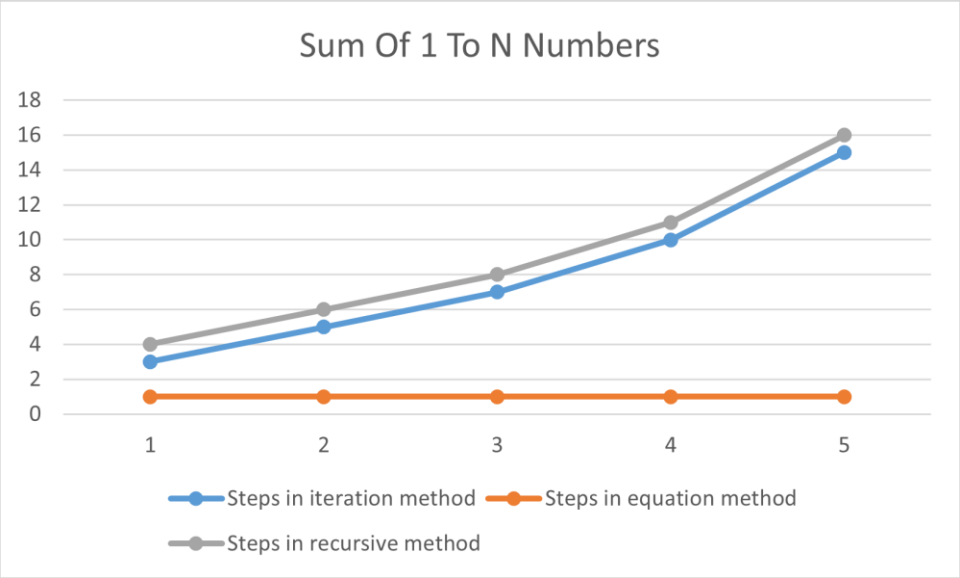
```
Vatsal …\DAA\Prac1 main !? v6.3.0 23:26 .\oneToN.exe
Give Me Number To Sum of Numbers from 1 to N You want Sum of: 5

----------------------------------------
Enroll Number: 142
Loop Method
Sum of 1 to 5 Number is: 15
Count of Step of Algorithm is : 5
----------------------------------------
----------------------------------------
Enroll Number: 142
Equation Method
Sum of 1 to 5 Number is: 15
Count of Step of Algorithm is : 1
----------------------------------------
----------------------------------------
Enroll Number: 142
Recursive Method
Sum of 1 to 5 Number is: 15
Count of Step of Algorithm is : 5
----------------------------------------
```

Analysis:

Value of N	Steps in iteration method	Steps in equation method	Steps in recursive method
3	3	1	3
5	5	1	5
7	7	1	7
10	10	1	10

15	15	1	15



Conclusion:

- The program compares iterative, recursive, and equation-based approaches to sum 1..N.
- The equation method finishes in O(1) time with only one counted step, while the loop and recursion grow with N.
- Recursion adds overhead and risks stack depth for large N, making it least practical here.
- Iteration is predictable but still linear, so it scales poorly compared to the constant-time formula.
- Choosing closed-form or more efficient algorithms reduces both runtime and step count.
- For bigger or similar problems, prioritize algorithmic efficiency before low-level optimizations.
- Measuring step counts highlights how the right algorithm dominates performance outcomes.