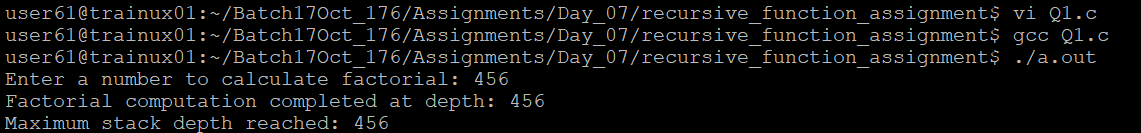
Recursive Function Assignment

1. WAP to calculate the maximum stack depth of a recursive call to a function. (For eg a factorial function ).

A screen shot of a computer program

Description automatically generated



2. What is tail recursion? Why is it important? Give an example

**Definition:**

Tail recursion occurs when the recursive call is the **last operation** performed in a function. This means that the result of the recursive call is directly returned without any further computation or processing.

**Why is it Important?**

Tail recursion is important because it allows the compiler or interpreter to optimize the recursion by reusing the current function's stack frame for the next recursive call. This optimization is called **Tail Call Optimization (TCO)**, which prevents stack overflow and reduces memory usage in deep recursion.

**Example of Tail Recursion:**

**Without Tail Recursion:**

int factorial(int n) {

if (n == 0 || n == 1)

return 1;

return n \* factorial(n - 1); // Multiplication occurs after the recursive call

}

**With Tail Recursion:**

int factorial\_tail(int n, int acc) {

if (n == 0 || n == 1)

return acc; // Return accumulated result

return factorial\_tail(n - 1, n \* acc); // Recursive call is the last operation

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

int result = factorial\_tail(num, 1); // Initial accumulator is 1

printf("Factorial of %d is %d\n", num, result);

return 0;

}

**Advantages of Tail Recursion:**

1. **Efficient Memory Usage:** Reuses the same stack frame, avoiding stack overflow.
2. **Improved Performance:** Often runs faster than non-tail-recursive functions when TCO is applied.

**Sample Output for Tail Recursion:**

**For input 5:**

Factorial of 5 is 120