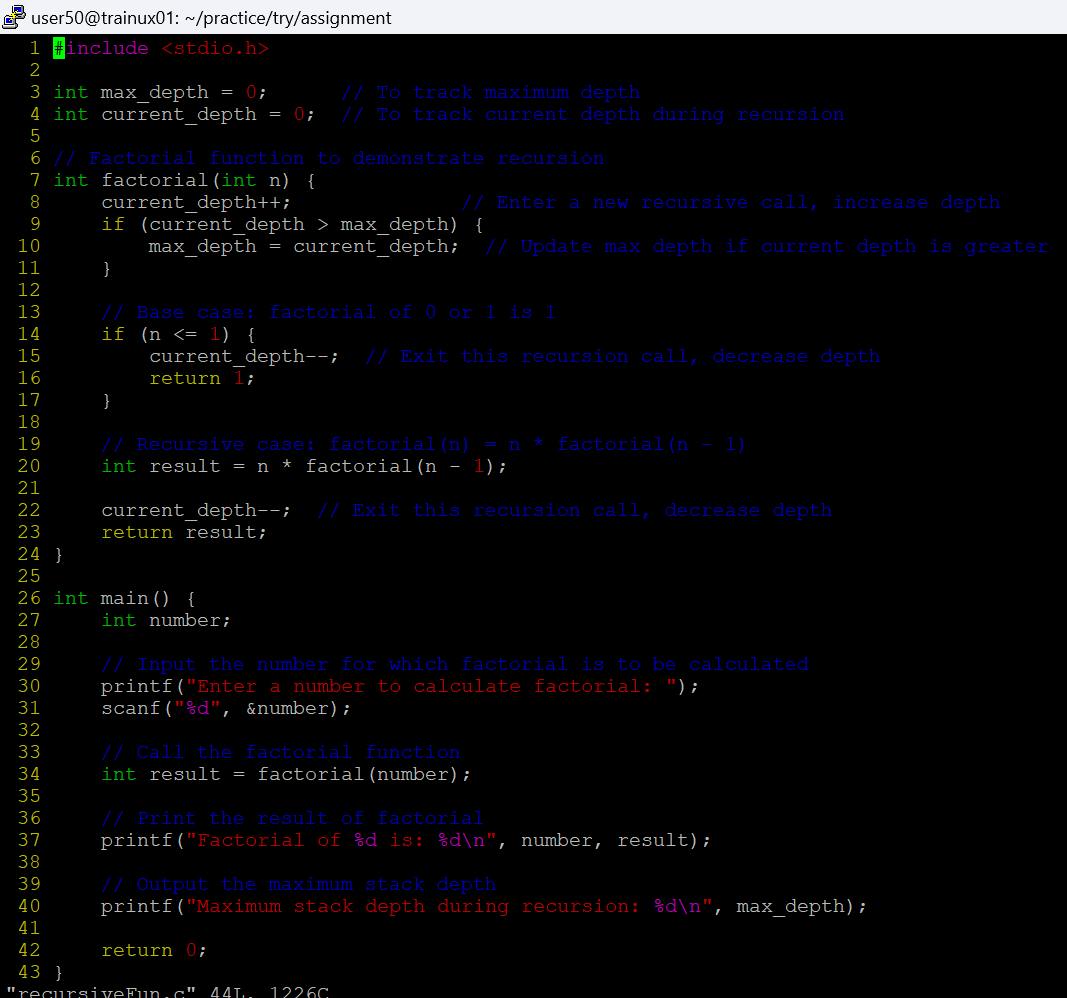
**Recursive Function Assignment**

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

**Tail recursion is a special form of recursion where the recursive call is the last operation in the function. In other words, in a tail-recursive function, the function calls itself at the very end of its execution, with no further computation or operations to be performed after the call.**

**In a normal recursive function, after making a recursive call, the function still needs to do some work (such as adding to a sum, or multiplying results) before returning the final result. However, in a tail-recursive function, the result of the recursive call is immediately returned, without needing any additional processing.**

**It is important because of:**

1. **Optimization**
2. **Memory efficiency**
3. **Performance**

**Example:**

**#include <stdio.h>**

**// Tail-recursive function to calculate factorial**

**int factorial\_tail\_recursive(int n, int accumulator) {**

**// Base case: if n is 0, return the accumulated result**

**if (n == 0) {**

**return accumulator;**

**} else {**

**// Tail recursive call**

**return factorial\_tail\_recursive(n - 1, n \* accumulator);**

**}**

**}**

**int main() {**

**int num = 5;**

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

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

**return 0;**

**}**

**Explanation:**

* **The function factorial\_tail\_recursive(int n, int accumulator) is defined to take two parameters:**
  + **n: The number whose factorial we want to calculate.**
  + **accumulator: The running product that accumulates the result during the recursion.**
* **Base case: When n == 0, the recursion stops, and the accumulator (which holds the factorial) is returned.**
* **Recursive case: The function makes a tail call to itself, passing n - 1 as the new value for n and n \* accumulator as the new value for the accumulator. This ensures that the recursive call is the last operation before returning the result.**