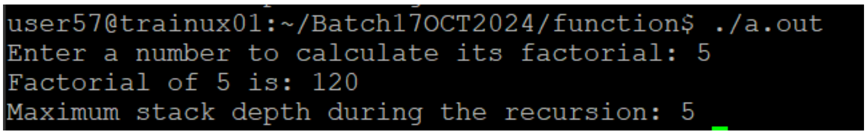
1. WAP to calculate the maximum stack depth of a recursive call to a function. (For eg a factorial function)

Sol:

```
#include <stdio.h>


static int current_depth = 0;

static int max_depth = 0;


int factorial(int n) {

  current_depth++;

  if (current_depth > max_depth) {

    max_depth = current_depth;

  }

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

    current_depth--;

    return 1;

  }

  int result = n * factorial(n - 1);

  current_depth--;

  return result;

}

int main() {

  int number;

  printf("Enter a number to calculate its factorial: ");

  scanf("%d", &number);

  int result = factorial(number);

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

  printf("Maximum stack depth during the recursion: %d\n", max_depth);

  return 0;

}
```

Output:



```
user57@trainux01:~/Batch17OCT2024/function$ ./a.out
Enter a number to calculate its factorial: 5
Factorial of 5 is: 120
Maximum stack depth during the recursion: 5
```

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

Sol:

Tail recursion is a special kind of recursion where the recursive call is the **last operation** in the function. In other words, a function is tail-recursive if it returns the result of the recursive call directly, without any further computation after the call.

Tail recursion is important because:

Memory Efficiency

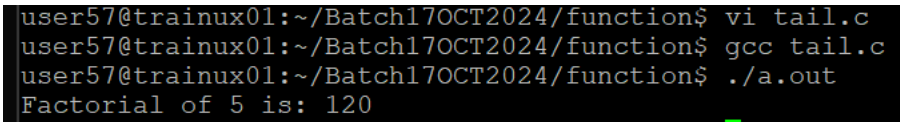
Avoid Stack Overflow

Optimization by Compiler

Example:

```
#include <stdio.h>

int factorial_tail_recursive(int n, int accumulator) {

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

    return accumulator;

  }

  return factorial_tail_recursive(n - 1, n * accumulator);

}

int main() {

  int number = 5;

  int result = factorial_tail_recursive(number, 1);

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

  return 0;

}
```

Output:



```
user57@trainux01:~/Batch17OCT2024/function$ vi tail.c
user57@trainux01:~/Batch17OCT2024/function$ gcc tail.c
user57@trainux01:~/Batch17OCT2024/function$ ./a.out
Factorial of 5 is: 120
```