# Tail Recursion

As we read before, that recursion is defined when a function invokes/calls itself.

**Tail Recursion**: A recursive function is said to be following Tail Recursion if it invokes itself at the end of the function. That is, if all of the statements are executed before the function invokes itself then it is said to be following Tail Recursion.

**Which one is Better-Tail Recursive or Non Tail-Recursive?**

The tail-recursive functions are considered better than non-tail recursive functions as tail-recursion can be optimized by the compiler. The idea used by compilers to optimize tail-recursive functions is simple, since the recursive call is the last statement, there is nothing left to do in the current function, so saving the current function's stack frame is of no use.

**Can a non-tail recursive function be written as tail-recursive to optimize it?**

->It is not possible for every non tail recursive function to be written as tail recursive.
->Consider the following function to calculate factorial of N. Although it looks like Tail Recursive at first look, it is a non-tail-recursive function. If we take a closer look, we can see that the value returned by **fact(N-1)** is used in **fact(N)**, so the call to fact(N-1) is not the last thing done by fact(N).

```
int fact(int N) :
    if (N == 0):
        return 1

    return N*fact(N-1)
```

The above function can be written as a Tail Recursive function. The idea is to use one more argument and accumulate the factorial value in the second argument. When N reaches 0, return the accumulated value.

```
int factTR(int N, int a) :
    if (N == 0):
        return a

    return factTR(N-1, N*a)
```

Dash
All
Articles
Videos
Problems
Quiz

Report An Issue

If you are facing any issue on this page. Please let us know.