Tail Recursion

XD

some random day

Tail Recursion

A recursive function is **tail-recursive**, when the final result of the recursive call is directly the result of the whole recursive function.

- **Not Tail-Recursive**:

```
sum [] = 0
sum (x:xs) = x + sum xs
```

- **Tail-Recursive**:

```
sum xs = help 0 xs
    where help acc [] = acc
          help acc (x:xs) = help (acc + x) xs
```

Tail Recursion

A tail recursive function can be optimized to avoid building up stack frames. Depenging on the support/option of the compiler, the result may vary.

Stack Frame

```
sum [] = 0
sum (x:xs) = x + sum xs
```

```
sum [5,4,3,2,1]
= 5 + sum [4,3,2,1]
= 5 + (4 + sum [3,2,1])
= 5 + (4 + (3 + sum [2,1]))
= 5 + (4 + (3 + (2 + sum [1])))
= 5 + (4 + (3 + (2 + (1 + sum []))))
= 5 + (4 + (3 + (2 + (1 + 0))))
= 5 + (4 + (3 + (2 + 1)))
= ...
= 15
```

Stack Frame

When Optimized:

```
sum xs = help 0 xs
    where help acc [] = acc
          help acc (x:xs) = help (acc + x) xs
```

```
sum [5,4,3,2,1]
= help 0 [5,4,3,2,1]
= help 5 [4,3,2,1]
= help 9 [3,2,1]
= help 12 [2,1]
= help 14 [1]
= help 15 []
= 15
```

Only When Optimized

You **must** turn on the option of the compiler in order for Tail Recursion being optimized. Otherwise it will behave just like normal functions.
Of course the compiler/executor itself must also support that. Java's JVM for example does not support this.

Tail Recursion in Haskell

When dealing with lazy evaluated languages, we can't just assume that code will behave the same way as it does in a strict language. In Haskell, the function call model is a little different, function calls might not use a new stack frame, so making a function tail-recursive typically isn't as big a deal.

Tail Recursion in C++

- **Not Tail-Recursive**:

```
int sum(int n) {
    if (n == 0) return 0;
    return n + sum(n - 1);
}
```

- **Tail-Recursive**:

```
int sum(int n, int acc = 0) {
    if (n == 0) return acc;
    return sum(n - 1, acc + n);
}
```

Tail Recursion in C++

When you turn on the optimization, both will be equivalent to:

```
int sum(int n) {
    int result = 0;
    for (int i = 0; i <= n; ++i) {
        result += i;
    }
    return result;
}
```

Yes, essentially they are just for-loops