realkinetic.com | @real_kinetic

REALKINETIC

YOUR POTENTIAL REALIZED

```haskell
-- This is a fold left of Ints
-- The left means we reduce (or traverse) from the left
foldlInt : (Int -> List Int -> List Int) -> Int -> List Int -> Int
foldlInt func aggVal list =
    case list of
        [] ->
            aggVal

        x :: xs ->
            foldl func (func x aggVal) xs
```

Ooh, we've got that "higher order functions" thing again

(A function that takes a function in this case)

```haskell
-- This is a fold left of Ints
-- The left means we reduce (or traverse) from the left
foldlInt : (Int -> List Int -> List Int) -> Int -> List Int -> Int
foldlInt func aggVal list =
    case list of
        [] ->
            aggVal

        x :: xs ->
            foldl func (func x aggVal) xs
```