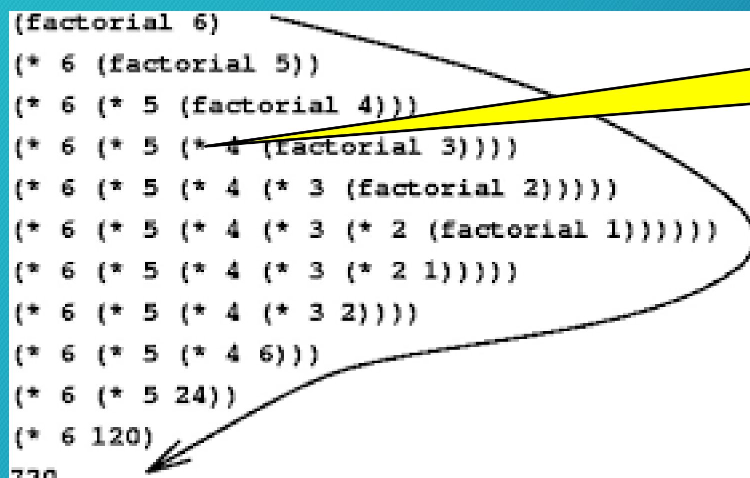
PROGRAMMING IN HASKELL

Chapter 6.2 – Tail Recursion

Recursion – why we need tail recursion


```
factRec :: Int -> Int
factRec 0 = 1
factRec n = n*factRec (n-1)
```



Pending ‘multiply’s

This is expensive and prone to stack overflows..

Tail recursion

- In tail recursion, the information that needs to be stored on the call stack is much smaller
- We use the base case in such a way that once it is called, the work is done (no pending operations)
- We use a helper function that in turn uses an accumulator

Example 1 – tail recursion


```
factTailRec ::Int -> Int
factTailRec n = helper n 1 where
    helper 1 acc = acc
    helper x acc = helper (x-1) (acc*x)
```

```
factTailRec 4
        = helper 4 1
        = helper 3 (4*1)
        = helper 2 (3 * 4)
        = helper 1 (2 *12)
```

```
= 24
```

General template

Some updated value

```
foo :: Int -> Int
foo x = foohelper x initial solution
where foohelper
      | base case = solution (based on accum)
   | otherwise = fooHelper (n-1)(updated accum)
```

Example 2 – tail recursion


```
sumInts :: [Int] -> Int
sumInts [] = 0
sumInts (n:ns) = n + sumInts ns
```

e.g. sumInts [1,3,5] = 9

We will rewrite this using tail recursion.

Exercise: Convince yourself that it is not currently tail recursive

Example 2 – tail recursion


```
sumIntsTail :: [Int] -> Int
sumIntsTail ns = helper ns 0 where
    helper [] acc = acc
    helper (n:ns) acc = helper ns (acc + n)
```

```
sumIntsTail [1,3,5]
      = helper [1,3,5] 0
      = helper [3,5] (1+0)
      = helper [5] (3+1)
      = helper [] (5+4)
= 9
```

Example 3 – tail recursion

myAdd takes two integers and returns their sum.
Note : we can only use +1 and -1

```
myAdd :: Int -> Int -> Int
myAdd x 0 = x
myAdd 0 y = y
myAdd x y = myAdd (x-1) (y+1)
```

Is this tail recursive?

Example 3 – tail recursion


```
myAdd :: Int -> Int -> Int
myAdd x 0 = x
myAdd 0 y = y
myAdd x y = myAdd (x-1) (y+1)
```

Is this tail recursive?

```
  myAdd 3 4
= myAdd 2 5
= myAdd 1 6
= myAdd 0 7
```

= 7

No pending ops after base case is called

This is tail recursive


ANY
QUESTIONS?