# Tutorium Funktionale Programmierung 2019

## Part 12 - different types of recursion, infinite lists and laziness

**VO - Part 7**

Benedikt Dornauer, 10.01.2020

Tutorium will take place on 21.01.2020 at HSB 4

# Exercise 12.1: **Mutual recursion** – What is the intention of this function?

```
f :: Int -> Bool
f 0 = True
f n = g (n - 1)

g :: Int -> Bool
g 0 = False
g n = f (n - 1)
```

# Recursion (without tail recursion)

fac 5 =
5 * ( fac 4) =
5 * ( 4 * (fac 3)) =
5 * ( 4 * ( 3 * (fac 2))) =
5 * ( 4 * ( 3 * (2 * (fac 1 ))) =
5 * ( 4 * ( 3 * ( 2 * ( 1 )))) =
5 * ( 4 * ( 3 * ( 2 ))) =
5 * ( 4 * ( 6 )) =
5 * ( 24 ) =
5 * ( 24 ) =
120

# Tail Recursion

recursion + **last** recursive call of function returns **the final result**

```
fac 5 =
5 * ( fac 4) =
5 * ( 4 * (fac 3)) =
5 * ( 4 * ( 3 * (fac 2))) =
5 * ( 4 * ( 3 * (2 * (fac 1 ))) =
5 * ( 4 * ( 3 * ( 2 * ( 1 )))) =
5 * ( 4 * ( 3 * ( 2 ))) =
5 * ( 4 * ( 6 )) =
5 * ( 24 ) =
5 * ( 24 ) =
120
```

VS

```
fac =
fac (1,5) =
fac ( 5, 4) =
fac ( 20 , 3) =
fac ( 60 , 2) =
fac ( 120 , 1) =
120
```

```haskell
rec1 :: Integer -> Integer
rec1 n
 | n == 1 = 1
 | otherwise = n * (rec1 (n-1))

recTail :: (Integer, Integer) -> Integer
recTail (n,x)
 | n == 1 = x
 | otherwise = recTail ( n-1 , x*n )

rec2 :: Integer -> Integer
rec2 n = recTail (n,1)
```

# Exercise 12.2: Tail-Recursion

The **Euler's totient function** is expressed in the following way

$$\varphi(n) := \left|\{a \in \mathbb{N} \mid 1 \leq a \leq n \wedge \mathrm{ggT}(a, n) = 1\}\right|$$

$|\,|$ … number of elements

Express the **Euler's totient function** using **tail-recursion.**

# Infinite Lists in Haskell

- Lists can be infinite ( ~> laziness)

```
f xs = map (*2) xs
```

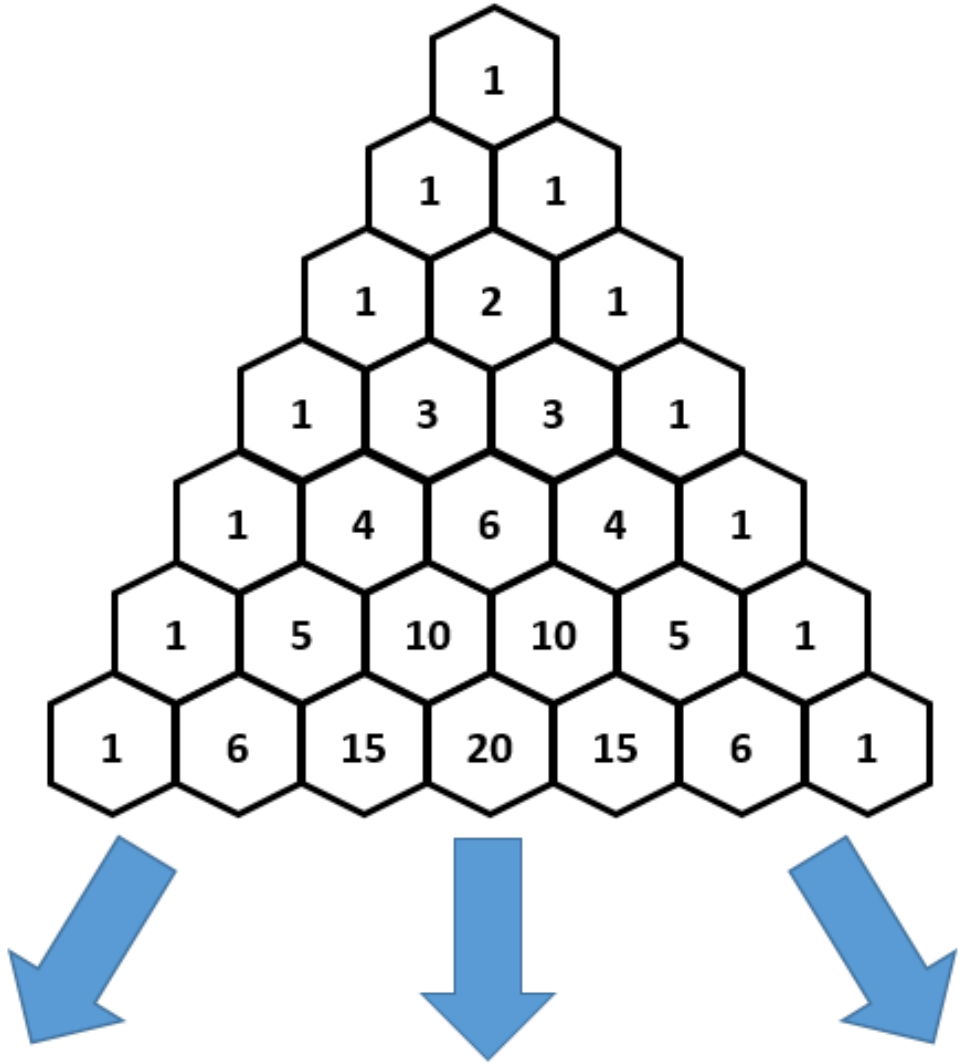
```
f [1..]
```

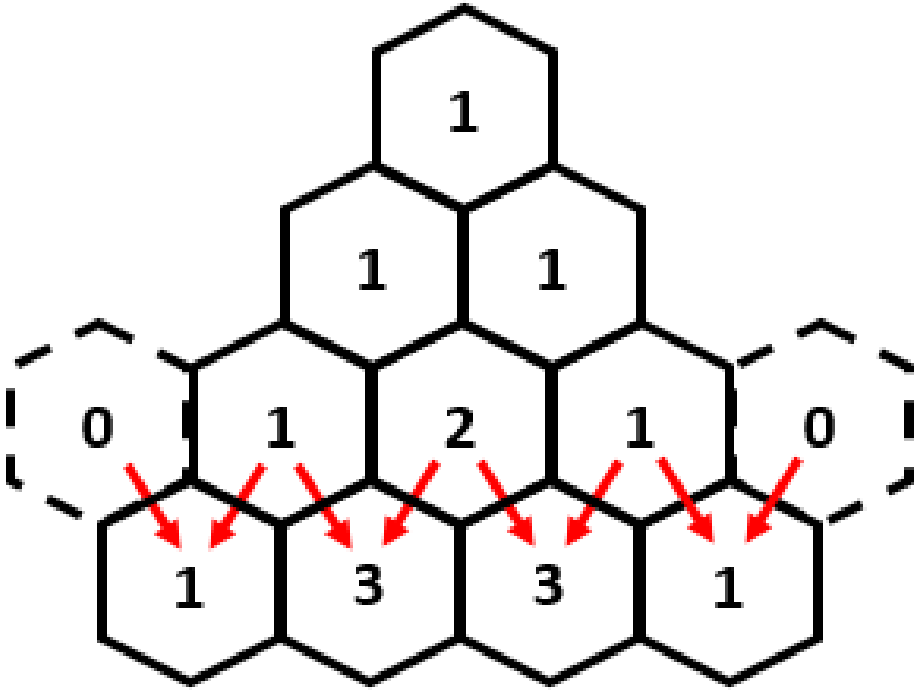
[2,4,6,8,10,12,14,16,18,20,2
2,24,26,28,30,32,34,36,38,40
,42,44,46,48,50,52,54,56,58,
60,62,64,66,68,70,72,74,76,7
8,80,82,84,86,88,90,92,94,96
,98,100,102,104,106,108,110

...

STRG + C

# Exercise 12.3: **Infinite List**

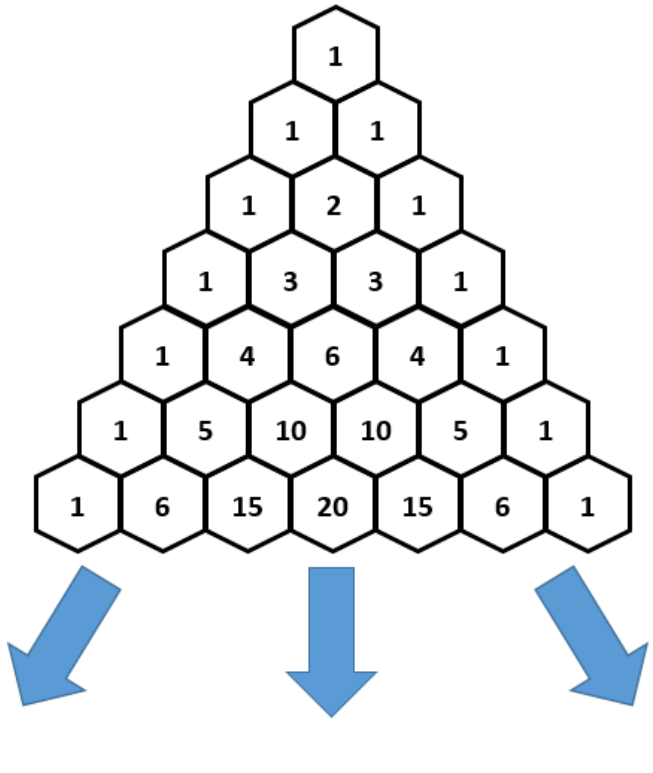



Img. : https://brilliant.org/wiki/pascals-triangle/

# Exercise 12.3: Infinite List

Try to define your own infinite list of „rows “ of the pascal triangle“.




[[1,1],
[1,2,1],
[1,3,3,1],
[1,4,6,4,1],
[1,5,10,10,5,1],
[1,6,15,20,15,6,1], …

Img. : https://brilliant.org/wiki/pascals-triangle/

# Exercise 12.4: Infinite List and Laziness

```haskell
echoes1 :: [Int] -> [Int]
echoes1 = foldr (\ x xs -> (replicate x x) ++ xs) []

echoes2 :: [Int] -> [Int]
echoes2 = foldl (\ xs x -> xs ++ (replicate x x)) []
```

1. What is the difference between echoes1 and echoes2 ?
2. What will happen if you try
   - take 10 (echoes1 [1..])
   - take 10 (echoes2 [1..])

src.: https://en.wikibooks.org/wiki/Haskell/Lists_III#Are_foldl_and_foldr_opposites?

# Questions? Need help? Feedback? etc.

- benedikt.dornauer@student.uibk.ac.at