Solution  Functional Programming

Instructions:

Solutions of the exercises are to be delivered before Thursday, the 15th of March at 10:15AM. Solutions should be placed in a separate folder with the name “Assignment03”. Please submit answers to all the exercises in one .hs file named “assignment03.hs”. Please use the provided template in which all the solutions should be written.

Exercise 1 (1.5 points)

Define a function firstNCatalan n in Haskell that will calculate and return as the result the list which contains the first n Catalan numbers. Catalan numbers are calculated based on the formula 

\[ C_n = \frac{(2n)!}{(n+1) n!}, n \geq 0. \]

Answer:

```haskell
fac n = case n of
    0 -> 1
    _ -> n * fac (n-1)
catalan n = if n >= 0 then fac (2*n) / (fac n * fac (n+1)) else 0
firstNCatalan n = [catalan x |x <-[0..n]]
```

Exercise 2 (1.5 points)

Define a function perfectNumbers n m in Haskell that returns as the result the list of all perfect numbers greater than n and smaller than m. A positive integer is perfect if it is equal to the sum of its proper positive factors.

Answer:

```haskell
factors n = [x | x <- [1..n-1], mod n x == 0 ]
isPerfect n = sum (factors n) == n
perfectNumbers n m = [x | x <- [n+1..m-1], isPerfect x]
```

Exercise 3 (1.5 points)

Define a function insert i n l in Haskell that returns as the result the list that contains as the first \(i\) elements the same ones as in the list \(l\), preserving the order, followed by the element \(n\) on the \(i\)-th position, and the remaining elements of the list \(l\), preserving the order. In case that \(i\) exceeds the size of the list, the resulting list should have all the elements of the list \(l\), preserving the order, and the element \(n\) as the last one. The index counting starts from zero.
Answer:
insert n [] = [n]
insert 0 n l = n:l
insert i n (x:xs) = x : insert (i-1) n xs

Exercise 4 (1.5 points)

Define a function indexes n l in Haskell that returns as the result the list containing all the indexes in the list l where the element n appears. In case that n is not contained in the list, the function returns an empty list. The index counting starts from zero.

Answer:
indexes a l = indexesFrom 0 a l
indexesFrom i n [] = []
indexesFrom i n (x:xs)
    | x == n = i : indexesFrom (i+1) n xs
    | otherwise = indexesFrom (i+1) n xs