FP Lab 5

1. (a) Define a function `add :: Nat -> Nat -> Nat` that computes the sum of two natural numbers using pattern-matching and recursion.

   (b) Define a function `mult :: Nat -> Nat -> Nat` that computes the product of two natural numbers using pattern-matching, recursion and the function `add` defined above.

2. Consider the following type of binary trees of integers:

   ```
   data Tree = Leaf Int | Node Tree Tree
               deriving (Show)
   ```

   Such a tree is *balanced* if the number of leaves in the left and right subtrees of every node differs by at most one, with leaves themselves being trivially balanced.

   Define a function `balanced :: Tree -> Bool` that decides whether a tree is balanced. (Hint: first define a function that returns the number of leaves of a tree.)

3. Define a function `balance :: [Int] -> Tree` that converts a non-empty list of integers into a balanced tree. (Hint: by using the function `splitAt :: Int -> [a] -> ([a], [a])`, first define a function that splits a list into two halves whose lengthes differ by at most one.)

4. (Hard and optional) *Nim* is a game that is played by two players on a board comprising five numbered rows of stars, which is initially set up as follows:

   ```
   1: * * * * *
   2: * * * *
   3: * * *
   4: * *
   5: *
   ```

   Two players take it in turn to remove one or more stars from the end of a single row. The winner is the player who removes the last star or stars from the board.

   Implement the game of *nim* in Haskell. (Hint: Represent the board as a list of five integers that give the number of stars remaining on each row. For example, the initial board is `[5,4,3,2,1]`.)