Assignment 1

1. Write a function

    ```
    fib :: Integer -> Integer
    ```

    for the Fibonacci sequence:

    $$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(n+2) &= f(n+1) + f(n) \quad \text{for } n \geq 0 \end{aligned}$$

    It does not have to be efficient.

2. Write a function

    ```
    prodlist :: [Integer] -> Integer
    ```

    that computes the product of the numbers in a list. E.g.,

    ```
    prodlist []
    prodlist [1,3,4]
    ```

    should be 1 and 12 respectively.

3. Write a function

    ```
    oddity :: [Int] -> [Bool]
    ```

    that scans the input list N of numbers, checks each one if it is even or odd, and returns a boolean list B of the same length in which each element is true iff the correspond element in N (by position) is odd. Examples:

    ```
    oddity [] = []
    oddity [1,2,3,5] = [true, false, true, true]
    ```

4. Modify the `Shape` type in the lecture to include two more shapes: triangle with three vertices, and polygon with a list of vertices. Each vertex is an ordered pair of floats, i.e., `(Float, Float)`.

    To avoid cluttering, you may use *type synomyn* in Haskell:

    ```
    type Vertex = (Float, Float)
    ```

    Then wherever you would write `(Float, Float)` you may write `Vertex` instead, and vice versa.

    Modify the area function accordingly.