Assignment 1

1. Write a function

   \[\text{fib} :: \text{Integer} \rightarrow \text{Integer}\]

   for the Fibonacci sequence:

   \[
   \begin{align*}
   f(0) &= 1 \\
   f(1) &= 1 \\
   f(n + 2) &= f(n + 1) + f(n) \quad \text{for } n \geq 0
   \end{align*}
   \]

   It does not have to be efficient.

2. Write a function

   \[\text{prodlist} :: [\text{Integer}] \rightarrow \text{Integer}\]

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

   \[
   \begin{align*}
   \text{prodlist} \, [] &= 1 \\
   \text{prodlist} \, [1,3,4] &= 12
   \end{align*}
   \]

   should be 1 and 12 respectively.

3. Write a function

   \[\text{oddity} :: [\text{Int}] \rightarrow [\text{Bool}]\]

   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:

   \[
   \begin{align*}
   \text{oddity} \, [] &= [] \\
   \text{oddity} \, [1,2,3,5] &= [\text{true}, \text{false}, \text{true}, \text{true}]
   \end{align*}
   \]

4. Modify the \texttt{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., \((\text{Float}, \text{Float})\).

   To avoid cluttering, you may use \textit{type synonym} in Haskell:

   \[\text{type Vertex} = (\text{Float}, \text{Float})\]

   Then wherever you would write \((\text{Float}, \text{Float})\) you may write Vertex instead, and vice versa.

   Modify the area function accordingly.