Inductive Types and Recursive Functions

Task 1
Augment the Shape type from lecture 6 with a constructor Star for \( n \)-pointed stars, where an \( n \)-pointed star of length \( l \) and height \( h \) consists of an \( n \)-sided regular polygon of face length \( l \) with an isosceles triangle of base \( l \) and height \( h \) attached along each face.

Task 2
Update the area function to be compatible with your new definition of Shape.

Type Constructors

Task 3
Write the following function, which returns the element at the specified index of a List, if any:

\[
\text{indexList} : (\text{index} : \text{Nat}) \to \text{List} \ a \to \text{Maybe} \ a
\]

Task 4
Write the following function, which returns the element at the specified index of a Vect:

\[
\text{indexVect} : (\text{index} : \text{Fin} \ n) \to \text{Vect} \ n \ a \to a
\]

Why do we not need Maybe in the return type?

Higher-Order Functions

Task 5
Write a zip function for trees:

\[
\text{zipTree} : (a \to b \to c) \to \text{Tree} \ a \to \text{Tree} \ b \to \text{Tree} \ c
\]

Task 6
Write the fold function for the parameterized type Maybe a.

Task 7
Use your fold for Maybes in order to write the map for Maybes as a one-liner:

\[
\text{mapMaybe} : (a \to b) \to \text{Maybe} \ a \to \text{Maybe} \ b
\]
Task 8
Suppose that we have a number of computations, each of type \( \text{IO (Either error Unit)} \), which when run may yield either the result \( \text{Right ()} \) if they complete normally or else \( \text{Left e} \), where \( e \) is an element of some type \( \text{error} \), if something goes wrong. Write a function that takes a list of such computations and returns a computation that tries to run them in order, but stops if it encounters an error, returning the error and discarding any pending computations from the list:

\[
\text{tryIOs} : \text{List (IO (Either error Unit))} \rightarrow \text{IO (Maybe error)}
\]

Task 9
Suppose that we again want to run our list of computations in order, but now we want to run them all unconditionally and return a list of any errors that occurred:

\[
\text{batchIOs} : \text{List (IO (Either error Unit))} \rightarrow \text{IO (List error)}
\]