Homework 3

Functional Programming (ITI0212)

due 2020.05.08

Problem 1
Write a default `Eq` implementation for shapely trees so that two trees are equal according to the `(==)` method just in case they have the same label at each node position.

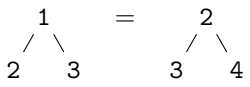
Problem 2
Recall the `Bifunctor` interface from lecture 9:

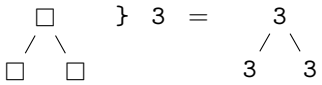
```
interface  Bifunctor (t : Type -> Type -> Type)  where
  bimap  :  (a -> b) -> (c -> d) -> t a c -> t b d
```

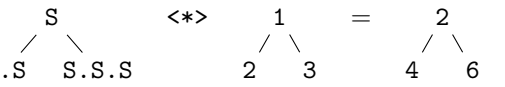
Write a default `Bifunctor` implementation for the `Either` type constructor so that:

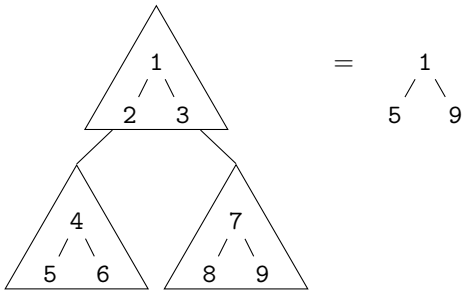
```
bimap S not (Left 3)  =  Left 4
bimap S not (Right False)  =  Right True
```

Problem 3
Write `Functor`, `Applicative`, and `Monad` instances for shapely trees, such that:

- `map S` 

- `pure {shape =` 

- 

- `join` 

Hint: The following functions will be helpful to implement `join`:

```
head  :  Tree (NodeShape l r) a -> a
head (Node left this right)  =  this
```

```
left_tail  :  Tree (NodeShape l r) a -> Tree l a
left_tail (Node left this right)  =  left

right_tail  :  Tree (NodeShape l r) a -> Tree r a
right_tail (Node left this right)  =  right
```

Problem 4
Convince Idris that addition is injective in both its right and left arguments:

```
plus_inj_right  :  {m , n_0 , n_1 : Nat} -> m + n_0 = m + n_1 -> n_0 = n_1

plus_inj_left   :  {m , n_0 , n_1 : Nat} -> n_0 + m = n_1 + m -> n_0 = n_1
```

Hint: It's easiest to write these in the order specified. You may also find the following easy lemma useful:

```
pred_equal  :  {m , n : Nat} -> S m = S n -> m = n
```

Problem 5
Convince Idris that if the sum of two numbers is even then the sum of the same two numbers in the opposite order is also even:

```
even_plus_sym  :  {m , n : Nat} -> Even (m + n) -> Even (n + m)
```

Problem 6
Write a function that cyclically permutes the elements of a vector by a given number of positions:

```
rotate_vect  :  Nat -> Vect n a -> Vect n a
```

Your function should behave as follows:

```
rotate_vect 0 [1 , 2 , 3]  =  [1 , 2 , 3]
rotate_vect 1 [1 , 2 , 3]  =  [2 , 3 , 1]
rotate_vect 2 [1 , 2 , 3]  =  [3 , 1 , 2]
rotate_vect 3 [1 , 2 , 3]  =  [1 , 2 , 3]
the (Vect _ Bool) (rotate_vect 42 [])  =  []
```