Exercises

1. Define the Idris function `(!!)` for extracting the nth element from a list (use zero for first element).

\[
(!!) : \{a : \text{Type}\} \rightarrow \text{List } a \rightarrow \text{Nat} \rightarrow \text{Maybe } a \\
[] \quad !! \ _ = \text{Nothing} \\
(x :: xs) \quad !! \ z = \text{Just } x \\
(_ :: xs) \quad !! \ (S \ n) = xs \quad !! \ n
\]

For CS 581: All functions in Idris should be total!
4. Define the Idris functions \( \text{eqNat} \) and \( \text{eqList} \) for comparing two natural numbers and two lists of values.

\[
\text{eqNat} : \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Bool} \\
\text{eqNat} \ Z \ Z = True \\
\text{eqNat} \ (S \ n) \ (S \ m) = \text{eqNat} \ n \ m \\
\text{eqNat} \ _ \ _ = False
\]

\[
\text{data Nat : Type where} \\
\text{Z : Nat} \\
\text{S : Nat} \rightarrow \text{Nat}
\]

\[
\text{eqList} : \text{Eq} \ a \Rightarrow \text{List} \ a \rightarrow \text{List} \ a \rightarrow \text{Bool} \\
\text{eqList} \ [] \ [] = True \\
\text{eqList} \ (x::xs) \ (y::ys) = x==y \land \text{eqList} \ xs \ ys \\
\text{eqList} \ _ \ _ \ _ = False
\]
5. Define the Idris function `eqVect` for comparing two vectors of values.

```
eqVect : Eq a => Vect n a → Vect n a → Bool
eqVect []      []      = True
eqVect (x::xs) (y::ys) = x==y && eqVect xs ys
```

Types make 3rd case impossible and thus unnecessary.
6. Define the functions \texttt{head} and \texttt{tail} for vectors.

\texttt{head} : \texttt{Vect (S n) a} \rightarrow \texttt{a}
\texttt{head} (x :: _) = x

\texttt{tail} : \texttt{Vect (S n) a} \rightarrow \texttt{Vect n a}
\texttt{tail} (_ :: xs) = xs

\texttt{data Vect : Nat} \rightarrow \texttt{Type} \rightarrow \texttt{Type} \texttt{where}
\texttt{Nil : Vect Z a}
\texttt{(_ ::) : a} \rightarrow \texttt{Vect n a} \rightarrow \texttt{Vect (S n) a}
Exercises

7. Define a type for matrices with \( n \) rows and \( m \) columns using nested vectors.

\[
\text{Matrix} : \text{Nat} \to \text{Nat} \to \text{Type} \to \text{Type} \\
\text{Matrix} \ n \ m \ a = \text{Vect} \ n \ (\text{Vect} \ m \ a)
\]

8. Define the functions \text{firstRow} and \text{firstCol} for matrices.

\[
\text{firstRow} : \text{Matrix} \ (S \ n) \ m \ a \to \text{Vect} \ m \ a \\
\text{firstRow} \ (xs :: \_) = xs
\]

\[
\text{firstCol} : \text{Matrix} \ n \ (S \ m) \ a \to \text{Vect} \ n \ a \\
\text{firstCol} \ xss = \text{map} \ \text{head} \ xss
\]
9. Define an Idris function `lift` for lifting an arbitrary `Nat → Nat` function to work on singletons.

Value argument used in result type

```
lift : (f : Nat → Nat) → Singleton n → Singleton (f n)
lift _ _ = The
```
10. **Draw the type partition defined by the** Even **data type.**

```
data Even : Bool → Type where
  E0  : Even True
  E1  : Even False
  ESS : Even b → Even b
```
Exercises

11. Draw instances of indexed types for the type \((=)\).
12. Verify the $S \ k$ case of the lemma.

lemma \{n=S \ k\} = cong lemma \{n=k\}

Inductive Hypothesis

Definition of \( \text{lemma} \)

Definition of \( \text{cong} \) with \( S \) for \( f \)

Substituting \( S \ k \) for \( n \)