1. Define the property \( \text{Len} \) (length of a list) as an Idris data type.

\[
\text{data Len : List a \rightarrow Nat \rightarrow Type where} \\
\text{Empty : Len [] 0} \\
\text{NonEmpty : Len xs n \rightarrow Len (x::xs) (S n)}
\]

2. Prove the following theorem about \( \text{Len} \).

\[
\text{lenAdd : Len xs n \rightarrow Len ys m \rightarrow Len (xs++ys) (n+m)} \\
\text{lenAdd \{xs=[]\} \ Empty q = q} \\
\text{lenAdd \{xs=z::zs\} \ (NonEmpty p) q = NonEmpty \ (lenAdd \{xs=zs\} p q)}
\]
3. **Prove the lemma** LenS.

```agda
lenS : Len xs n → Len (x∷xs) (S n)
lenS = NonEmpty
```

4. **Prove the lemma** LenSF.

```agda
lenSF : Flen xs = n → Flen (x∷xs) = S n
lenSF p = rewrite p in Refl
```

**data** Len : List a → Nat → Type where
Empty : Len [] 0
NonEmpty : Len xs n → Len (x∷xs) (S n)

**Flen** : List a → Nat
Flen [] = 0
Flen (x∷xs) = 1 + Flen xs

**Normalization (def. of Flen)**

\[
p : Flen xs = n \\
Refl : Flen (x∷xs) = S n \\
Refl : 1 + Flen xs = S n \\
Refl : 1 + n = S n \\
Refl : S n = S n
\]

**Normalization (def. of +)**

\[
rewrite \ p \ in \ Refl
\]