Exercise sheet 2

Max Wisniewski, Alexander Steen

Task 1

We first prove a lemma that is being used in the induction proof of the task's proposition:
Lemma: `f (foldNat' f e n) =foldNat' f (f e) n`.
Proof by induction:
Base: `n = O`

```
    f (foldNat' f e O)
= {Def.}
    f e
= {Def. inverse}
    foldNat' f (f e) O
```

Step: `n` $\rightsquigarrow$ `S n`

```
    f (foldNat' f e (S n))
= {Def.}
    f (foldNat' f (f e) n)
= {Induction Hyp.}
    foldNat' f (f (f e)) n
= {Def. inverse}
    foldNat' f (f e) (S n)
```

□

Proposition: `foldNat f e =foldNat' f e`.
Proof by induction:
Base: `n = O`

```
    foldNat f e O
= {Def.}
    e
= {Def. inverse}
    foldNat' f e O
```

Step: `n` $\rightsquigarrow$ `S n`

```
    foldNat f e (S n)
= {Def.}
    f (foldNat f e n)
= {Induction Hyp.}
    f (foldNat' f e n)
= {Lemma}
    foldNat' f (f e) n
= {Def. inverse}
    foldNat' f e (S n)
```

□

Task 2

```
fact1 :: Nat → Nat
fact1 = paraNat ((uncurry  times) ∘ (first S)) (S O)

fact2 :: Nat → Nat
fact2 = snd ∘ (foldNat' (λ(x,y) → (S x, times (S x) y)) (O, S O))
```

Task 3

```
lengthL :: [a] → Nat
lengthL = foldr (λa b → S b) O

sumL :: [Nat] → Nat
sumL = foldr plus O

prodL :: [Nat] → Nat
prodL = foldr times (S O)
```

Task 4

```
paraList :: (b → ([b],a) → a) → a → [b] → a
paraList f e []       = e
paraList f e (x:xs)   = f x (xs, paraList f e xs)

headL :: [a] → Maybe a
headL =  paraList (λa _ → Just a) Nothing

tailL :: [a] → Maybe [a]
tailL = paraList (λ _ (as,_) → Just as) Nothing
```