```
Aufgabe 2
1.
data Nat = Zero |
           Succ Nat

add :: Nat -> Nat -> Nat
add x Zero = x
add x (Succ y) = Succ (add x y)

2.
IA:
Zeige Gültigkeit für z = Zero
add (add x y) Zero = add x y - nach Def 1
add x (add y zero) = add x y - nach Def 1

IV:
add (add x y) z = add x (add y z)

IS:
add x (add  y (Succ z)) = add x (Succ (add y z)) - nach Def 2
                        = Succ (add x (add y z)) - nach Def 2
                        = Succ (add (add x y) z) - nach IV
                        = add (add x y) (succ z) - nach Def 2

Aufgabe 3

1.
IA:
[] ++ [] = []

IV:...

IS:
els + [] = els => el:els ++ []
                = el:els
(el:els) + [] = el:els ++ []                (Def (++))
              = el:els                      (IV)

2.
IA:
length (L ++ []) = length L + length []

IV:
... (L ++ []) = ... + length []

IS:
length (el: (els ++ liste))
1 + length (els ++ liste)                   (Def length)
1 + length els + length liste               (IV)
= 1 + length (el:els) + length (liste)      (Def length)

3.
IA:
reverse L ++ reverse [] = reverse ([] ++ L)
reverse L = reverse L

IV:
reverse 1 ++ reverse 2 = reverse (2 ++ 1)

IS:
```

```
reverse 1 ++ reverse (el:els)
reverse 1 ++ reverse els ++ [el]             (Def reverse)
reverse (els ++ 1) ++ [el]                   (IV)
reverse (el:(els ++ 1))                      (Def reverse)
reverse (el:els) ++ 1                        (Def (++))

4.
IA:
reverse (reverse []) = []
[] = []                                      (Def reverse)

IV:
reverse (reverse liste) = liste

IS:
reverse (reverse el:els)
reverse (reverse els ++ [el])                (Def reverse)
reverse [el] ++ reverse (reverse els)        (Def 3. IV)
reverse [el] ++ els                          (IV)
reverse (el:[]) ++ els                       (Haksell)
reverse [] ++ [el] ++ els                    (Def reverse)
[] ++ [el] ++ els                            (Def reverse)
[el] ++ els                                  (Def (++))
(el:[]) ++ els                               (Haskohl)
el:([] ++ els)                               (Def reverse)
el:els                                       (Def (++))
```