```haskell
-- Model solution for Tut09
--
-- Copyright [2000..2004] Manuel M T Chakravarty

module Tut09
where

import Prelude hiding (abs, signum, (++))

abs                  :: Int -> Int
abs k | k >= 0       = k
      | otherwise    = -k

signum               :: Int -> Int
signum k | k > 0     = 1
         | k == 0    = 0
         | k < 0     = -1

{- -----------------------------

Property: abs n * signum n = n

PROOF:

Case n > 0:

    abs n * signum n
  = {Due to n > 0, abs.1}
    n * signum n
  = {Due to n > 0, signum.1}
    n * 1
  = {neutral of *}
    n

Case n = 0:

    abs 0 * signum 0
  = {abs.1}
    0 * signum 0
  = {0 * a = 0}
    0

Case n < 0:

    abs n * signum n
  = {Due to n < 0, abs.2}
    -n * signum n
  = {Due to n < 0, signum.3}
    -n * -1
  = {Arithmetic}
    n

QED
-}


(++)            :: [a] -> [a] -> [a]
[]      ++ ys   = ys
(x:xs) ++ ys   = x : (xs ++ ys)

{- -----------------------------

Property: P(xs) ===   (xs ++ ys) ++ zs = xs ++ (ys ++ zs)

PROOF:

Induction over xs.

Base: P([]) ===   ([] ++ ys) ++ zs = [] ++ (ys ++ zs)
```

```
 Left side:

    ([] ++ ys) ++ zs
  = {++.1}
    ys ++ zs

 Right side:

    [] ++ (ys ++ zs)
  = {++.1}
    ys ++ zs

Induction: P(x:xs) ===   ((x:xs) ++ ys) ++ zs = (x:xs) ++ (ys ++ zs)

 Left side:

    ((x:xs) ++ ys) ++ zs
  = {++.2}
    (x:(xs ++ ys)) ++ zs
  = {++.2}
    x:((xs ++ ys) ++ zs)
  = {Induction hypothesis}
    x:(xs ++ (ys ++ zs))

 Right side:

    (x:xs) ++ (ys ++ zs)
  = {++.2}
    x:(xs ++ (ys ++ zs))

QED
-}
```