```haskell
module Clase1
where

sumatoria1 :: Num a => [a] -> a
sumatoria1 [] = 0
sumatoria1 (x:xs) = x + sumatoria1 xs

sumatoria2 :: Num a => [a] -> a
sumatoria2 xs = foldr suma 0 xs
    where
      suma x y = x + y
-- Hugs.Base> :t foldr
-- foldr :: (a -> b -> b) -> b -> [a] -> b

-- Dar idea intuitiva de foldr (remplazo constructores por op)


sumatoria3 :: Num a => [a] -> a
sumatoria3 xs = foldr (+) 0 xs
-- Dar secciones


sumatoria4 :: Num a => [a] -> a
sumatoria4 = foldr (+) 0
-- Dar explicacion de tipos

productoria :: Num a => [a] -> a
productoria = foldr (*) 1

paraTodo :: [Bool] -> Bool
paraTodo = foldr (&&) True

existe :: [Bool] -> Bool
existe = foldr (||) False

largo1 :: Num a => [b] -> a
-- anda pero es raro para una primera clase
-- largo1 :: [a] -> Integer
largo1 = foldr mas1 0
         where
           mas1 x n = 1 + n

largo2 :: Num a => [b] -> a
largo2 = foldr ((+) . const 1) 0
-- Ver que prelude
-- const          :: a -> b -> a
-- const k _       = k
-- (.)            :: (b -> c) -> (a -> b) -> (a -> c)
-- (f . g) x       = f (g x)
-- Hacer reducciones de ((+) . const 1) x n

partir1 :: Ord a => a -> [a] -> ([a],[a])
partir1 p [] = ([],[])
partir1 p (x:xs) | x <= p = (x:ys, zs)
                 | x > p  = (ys, x:zs)
                where
                  (ys,zs) = partir1 p xs

partir2 :: Ord a => a -> [a] -> ([a],[a])
partir2 p = foldr elige ([],[])
    where
```

```haskell
      elige x (ys,zs) | x <= p = (x:ys, zs)
                      | x > p  = (ys, x:zs)

partir3 :: (a-> a -> Bool) -> a -> [a] -> ([a],[a])
partir3 f p = foldr elige ([],[])
    where
      elige x (ys,zs) | f x p = (x:ys, zs)
                      | otherwise = (ys, x:zs)
-- Insertion Sort
-----------------

insord :: Ord a => a -> [a] -> [a]
insord x [] = [x]
insord x (y:ys) | x <= y = x:y:ys
                | x > y  = y:insord x ys

ordena :: Ord a => [a] -> [a]
ordena [] = []
ordena (x:xs) = insord x (ordena xs)


-- Quick Sort
-------------

quicksort1 :: Ord a => [a] -> [a]
quicksort1 [] = []
quicksort1 (x:xs) = (quicksort1 (x:ys)) ++ (quicksort1 zs)
    where
      (ys,zs) = partir3 (<=) x xs
-- No anda!!

quicksort2 :: Ord a => [a] -> [a]
quicksort2 [] = []
quicksort2 (x:xs) = (quicksort2 ys) ++ [x] ++ (quicksort2 zs)
    where
      (ys,zs) = partir3 (<=) x xs
```