```haskell
module Clase2
where

-- 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)

-- Merge Sort
-------------

partirMedio1 :: [a] -> ([a],[a])
partirMedio1 xs = (take coc xs, drop coc xs)
                  where
                    coc = length xs ‘div‘ 2

partirMedio2 :: [a] -> ([a],[a])
partirMedio2 xs = splitAt (length xs ‘div‘ 2) xs


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

mergeSort :: Ord a => [a] -> [a]
mergeSort [] = []
mergeSort [x] = [x]
mergeSort xs = merge (mergeSort ys) (mergeSort zs)
    where
      (ys, zs) = partirMedio2 xs


-- Hanoi
--------

-- hanoi :: Integer -> Int -> Int -> [(Int,Int)]
hanoi :: Integral a => a -> Int -> Int -> [(Int,Int)]
hanoi 0 s t = []
hanoi (n+1) s t = hanoi n s i ++ [(s,t)] ++ hanoi n i t
                  where
                    i = 3 - (s+t)
```