Code (Tuesday Week 3)

```
import Data.List
import Test.QuickCheck
import Test.QuickCheck.Modifiers(OrderedList(..))

split :: [a] -> ([a],[a])
split [] = ([],[])
split [x] = ([x],[])
split (x:y:xs) = let (as, bs) = split xs
                 in (x:as, y:bs)


split_spec xs = let (as, bs) = split (xs :: [Int])
                in  permutation xs (as ++ bs)


-- leads to unreliable tests
permutation xs ys x = count x xs == count x ys
  where
    count x xs = length (filter (== x) xs)

-- more reliable definition
permutation' xs ys = sort xs == sort ys


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



mergeSpec1 xs ys = permutation (merge (xs :: [Int]) ys)
                               (xs ++ ys)

mergeSpec2 (Ordered xs) (Ordered ys) =
               sorted (merge (xs :: [Int]) ys)

sorted :: (Ord a) => [a] -> Bool
sorted [] = True
sorted [x] = True
sorted (x:y:zs) = x <= y && sorted (y:zs)
```

```haskell
unitTest = sort [4,3,2,1] == [1,2,3,4]

mergeSort :: (Ord a) => [a] -> [a]
mergeSort []  = []
mergeSort [x] = [x]
mergeSort ls  = let (as, bs) = split ls
                    as' = mergeSort as
                    bs' = mergeSort bs
                in merge as' bs'

mergeSortSpec1 xs = sorted (mergeSort (xs :: [Int]))

mergeSortSpec2 xs = permutation (mergeSort xs) (xs :: [Int])

mergeSortExtra xs = sort xs == mergeSort xs


main = do
  quickCheck mergeSortSpec1
  quickCheck mergeSortSpec2
```