Recursive Haskell!Recursive Haskell!Recursive Haskell!Recursive Haskell!

EECS 368 Haskell Programming Assignment 2

Due: Friday 4th May, 10am (start of class)

Define recursive function

merge :: Ord a => [a] -> [a] -> [a]

that merges two sorted lists to give a single sorted list.

```
> merge [2,5,6] [1,3,4]
[1,2,3,4,5,6]
```

• Test out your function with Haskell.
• You should assume that the input is sorted.
• You should not use any extra functions (like insert).
• Hand in your Haskell code, and your test cases. *You need test cases.* (This can be one piece of paper, if you wish).

```
leftside :: [a]->[a]
leftside xs = take (length xs `div` 2) xs

rightside :: [a]->[a]
rightside xs = drop (length xs `div` 2) xs

test cases:
   1) leftside [1,2,3,4,5]  = [1,2]
      rightside [1,2,3,4,5] = [3,4,5]
   2) leftside [1,2,3,4,5,6] = [1,2,3]
      rightside [1,2,3,4,5,6] = [4,5,6]
merge :: Ord a => [a] -> [a] -> [a]
merge xs [] = xs
merge [] ys = ys
merge (x:xs) (y:ys)
          | (x <= y)  = x:(merge xs (y:ys))
          | otherwise = y:(merge (x:xs) ys)
   3) test case:
   merge [1,6,9] [2,7,10]
      [1,2,6,7,9,10]
    merge [2,4,7,9] [0,3,4]
      [0,2,3,4,4,7,9]
      merge [6,8,11] [0,3,4,5]
      [0,3,4,5,6,8,11]
```