```haskell
-- Name: Xiaomeng Cao
-- Email: xcao07@syr.edu
-- Section: CST1-109

import Data.Char

locate :: Eq a => a -> [a] -> [Int]
locate x ys = map fst (filter ((==x).snd) (zip [1..] ys))

histogram :: [Int] -> String
histogram xs = concat (map f xs)
    where
      f :: Int -> String
      f x = (concat (replicate x "*")) ++ "\n"

manyFuns :: [a -> b] -> a -> [b]
manyFuns fs v = map ($ v) fs

mySort :: Ord a => (a -> a -> Bool) -> [a] -> [a]
mySort p [] = []
mySort p (x:xs) = ins p x (mySort p xs)
    where
      ins p y [] = [y]
      ins p y (z:zs)
          | p y z = y:z:zs
          | otherwise = z:(ins p y zs)

isFixPt :: Eq a => (a -> a) -> a -> Bool
isFixPt f val = (f val) == val

changeFirst :: (a -> Bool) -> a -> [a] -> [a]
changeFirst p val [] = []
changeFirst p val (x:xs)
    | p x = val : changeFirst p x xs
    | otherwise = x : changeFirst p val xs
```