```haskell
-- Gabriel Marques de Albuquerque

-- Q1

prswap :: [a] -> [a]
prswap [] = []
prswap (x:y:tail) = y:x: prswap (tail)



-- Q2
fml :: [a] -> (a, a, a)
fml l = (frst, mid, last)
   where
      frst = head l
      mid = head (drop ((div (length l) 2)) l)
      last = head (drop ((length l)-1) l)

-- Q3
divide :: Int -> Int -> Bool
divide i j = mod i j == 0

calcPrimes :: [Int]
calcPrimes = [num | num <- [2..], all (\d -> not (divide num d)) [2..num-1]]


-- Q4
map' :: (a->b) -> [a] -> [b]
map' f xs = foldr (fAccumulative f) [] xs
   where
      fAccumulative f x accumulated = f x : accumulated

-- Q5

-- a)
merge :: Ord a => [a] -> [a] -> [a]
merge [] ys = ys
merge xs [] = xs
merge metade1@(x:xs) metade2@(y:ys)
   | y <= x = y: merge metade1 ys
   | otherwise = x: merge xs metade2

-- b)
metade :: [a] -> ([a], [a])
metade [] = ([], [])
metade [x] = ([x], [])
metade (xs) = splitAt ((length xs) `div` 2) xs

-- c)
msort :: Ord a => [a] -> [a]
msort [] = []
msort [x] = [x]
msort xs = merge (msort (fst (metade xs))) (msort (snd (metade xs)))


-- Q6

-- a)
-- Vazia -> pilha vazia
data Pilha t = Vazia | Pilha [t] deriving Show

-- b)
push :: t -> Pilha t -> Pilha t
push t Vazia = Pilha [t]
push t (Pilha x) = (Pilha (t:x))

pop :: Pilha t -> (t, Pilha t)
pop Vazia = error "Pilha vazia"
pop (Pilha (t:x)) = (t, Pilha x)
```