```haskell
-- Name: Xiaomeng Cao
-- Email: xcao07@syr.edu
-- Section: CST 1-019

import Data.Char

getInteger :: IO Integer
getInteger = do line <- getLine
                return (read line :: Integer)

getFloat :: IO Float
getFloat = do line <- getLine
              return (read line :: Float)

stats :: IO (Float, Float)
stats = do putStrLn ("Please enter three numbers: ")
           x <- getFloat
           y <- getFloat
           z <- getFloat
           return (sum[x,y,z]/3, sum[x,y,z]-maximum[x,y,z]-
minimum[x,y,z])

printVert :: String -> IO Int
printVert [] = do return (0)
printVert (c:cs) = do putChar (c)
                      putStrLn ("")
                      lines (cs)
                      return (length (c:cs))
                             where lines :: String -> IO ()
                                   lines []     = return ()
                                   lines (x:xs) = do
putChar (x)

putStrLn ("")
                                                        lines
(xs)
```

```
                                                        return
()

displayWords :: IO ()
displayWords = do putStr ("Please enter a line of text: ")
                  line <- getLine
                  let word = words line
                  lines word
                        where lines :: [String] -> IO()
                              lines []     = return()
                              lines (x:xs) = do putStrLn(x)
                                                lines xs
                                                return()
displayWords2 :: IO ()
displayWords2 = do putStr ("Please enter a line of text: ")
                   line <- getLine
                   let word = words line
                   lines 1 word
                         where lines :: Int -> [String] ->
IO ()
                               lines _ []     = return ()
                               lines n (x:xs) = do putStr
(show n ++ ". ")
                                                   putStrLn
(x)
                                                   lines
(n+1) xs
                                                   return
()

nonzeros :: IO [Integer]
nonzeros = do x <- getInteger
              if x == 0
                 then return []
                 else do y <- nonzeros
                         return (list x y)
```

```haskell
list :: Integer -> [Integer] -> [Integer]
list 0 [] = []
list x [] = [x]
list x y  = x:y

posAndNegs :: IO ()
posAndNegs = do putStrLn ("Please enter a series of
integers (0 to terminate ): ")
                x <- nonzeros
                let a = length (filter (>0) x ++ filter
(<0) x)
                putStrLn ("Number of nonzero values
entered: " ++ show a)
                let b = length (filter (>0) x)
                putStrLn ("Number of positive entered: " ++
show b)
                let c = minimum x
                if c < 0
                   then putStrLn ("Smallest negative number
entered: " ++ show c)
                   else putStrLn ("You did not enter any
negative numbers.")
                return ()
```