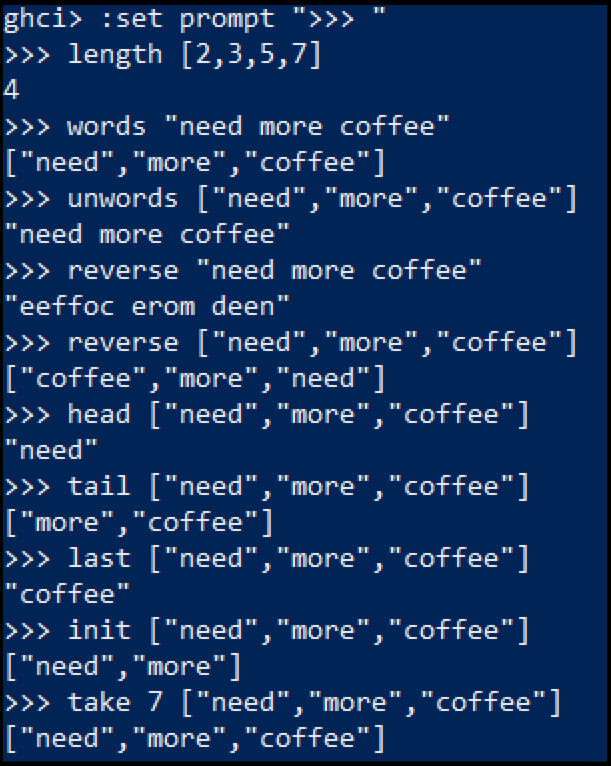
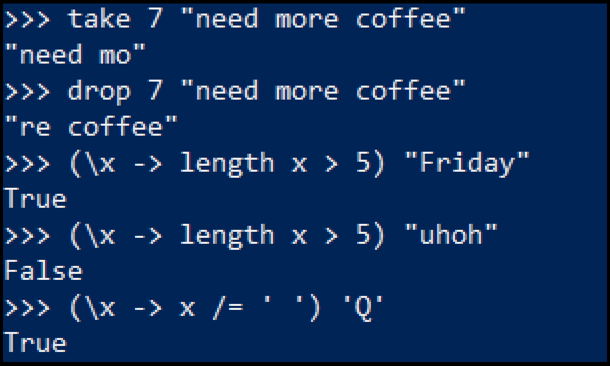
CSC 344 First Haskell Programming Assignment Solution

First Task: Mindfully Mimicking the Demo

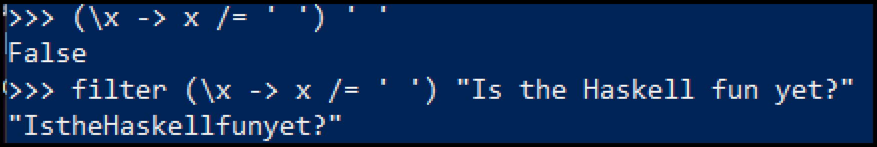
> Demo



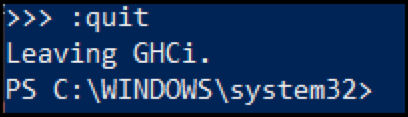
```
ghci> :set prompt ">>> "
>>> length [2,3,5,7]
4
>>> words "need more coffee"
["need","more","coffee"]
>>> unwords ["need","more","coffee"]
"need more coffee"
>>> reverse "need more coffee"
"eeffoc erom deen"
>>> reverse ["need","more","coffee"]
["coffee","more","need"]
>>> head ["need","more","coffee"]
"need"
>>> tail ["need","more","coffee"]
["more","coffee"]
>>> last ["need","more","coffee"]
"coffee"
>>> init ["need","more","coffee"]
["need","more"]
>>> take 7 ["need","more","coffee"]
["need","more","coffee"]
```



```
>>> take 7 "need more coffee"
"need mo"
>>> drop 7 "need more coffee"
"re coffee"
>>> (\x -> length x > 5) "Friday"
True
>>> (\x -> length x > 5) "uhoh"
False
>>> (\x -> x /= ' ') 'Q'
True
```



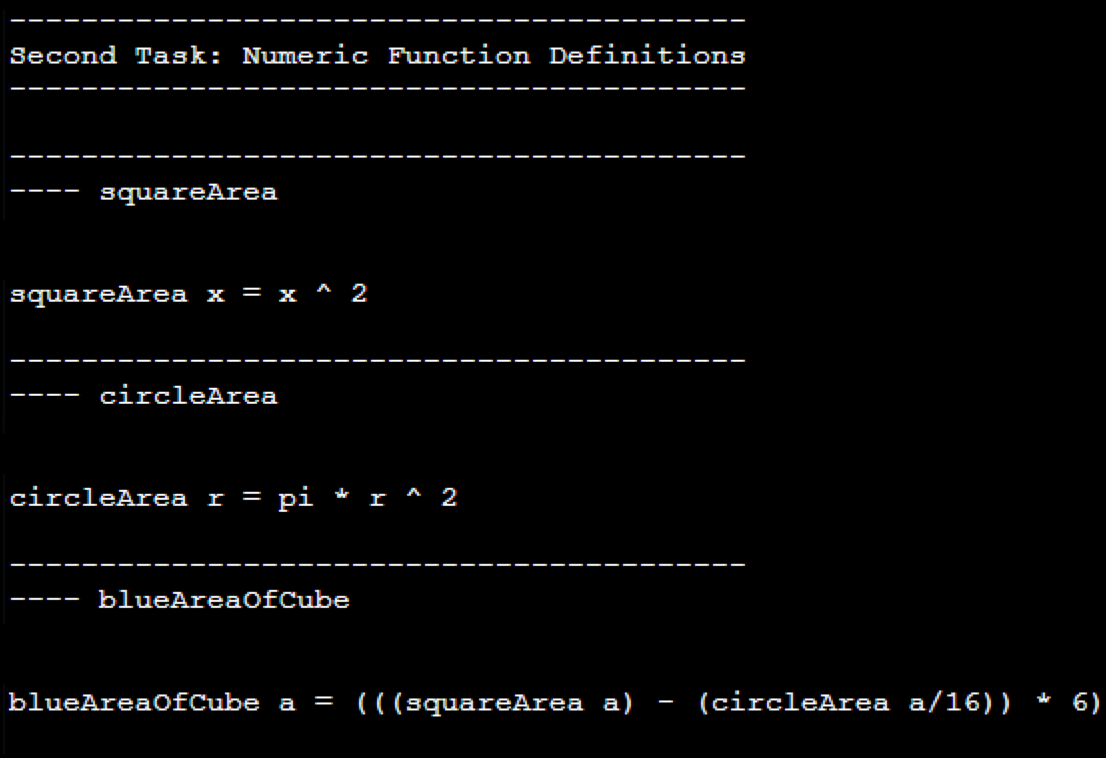
```
>>> (\x -> x /= ' ') ' '
False
>>> filter (\x -> x /= ' ') "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
```



```
>>> :quit
Leaving GHCi.
PS C:\WINDOWS\system32>
```

Second Task: Numeric Function Definitions

> Code



```
----------------------------------------
Second Task: Numeric Function Definitions
----------------------------------------

----------------------------------------
---- squareArea

squareArea x = x ^ 2

----------------------------------------
---- circleArea

circleArea r = pi * r ^ 2

----------------------------------------
---- blueAreaOfCube

blueAreaOfCube a = (((squareArea a) - (circleArea a/16)) * 6)
```

```
-----------------------------------------
---- paintedCube1


paintedCube1 n =
       if n > 2 then (6 * (n - 2) ^ 2)
       else 0

-----------------------------------------
---- paintedCube2


paintedCube2 n =
       if n > 2 then (12 * (n - 2))
       else 0
```

> Demo

Third Task: Puzzlers

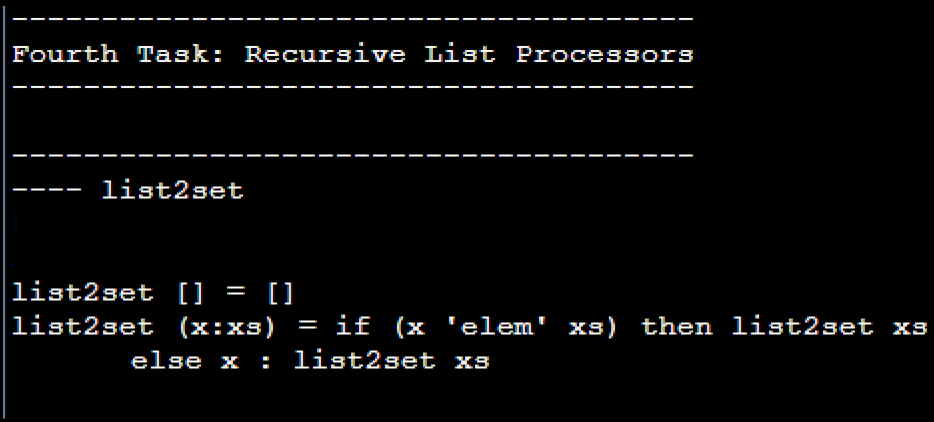
> Code

```
---------------------------------
Third Task: Puzzlers
---------------------------------

---------------------------------
---- reverseWords


reverseWords theWords = unwords(reverse(words theWords))

---------------------------------
---- averageWordLength


averageWordLength length =
       fromIntegral(sum(map length(words length))) /
       fromIntegral(length(words length))
```

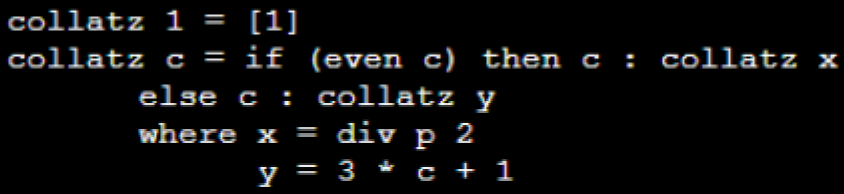
> Demo

Fourth Task: Recursive List Processors

> Code



```
--------------------------------------
Fourth Task: Recursive List Processors
--------------------------------------


--------------------------------------
---- list2set


list2set [] = []
list2set (x:xs) = if (x 'elem' xs) then list2set xs
       else x : list2set xs
```



```
collatz 1 = [1]
collatz c = if (even c) then c : collatz x
       else c : collatz y
       where x = div p 2
             y = 3 * c + 1
```