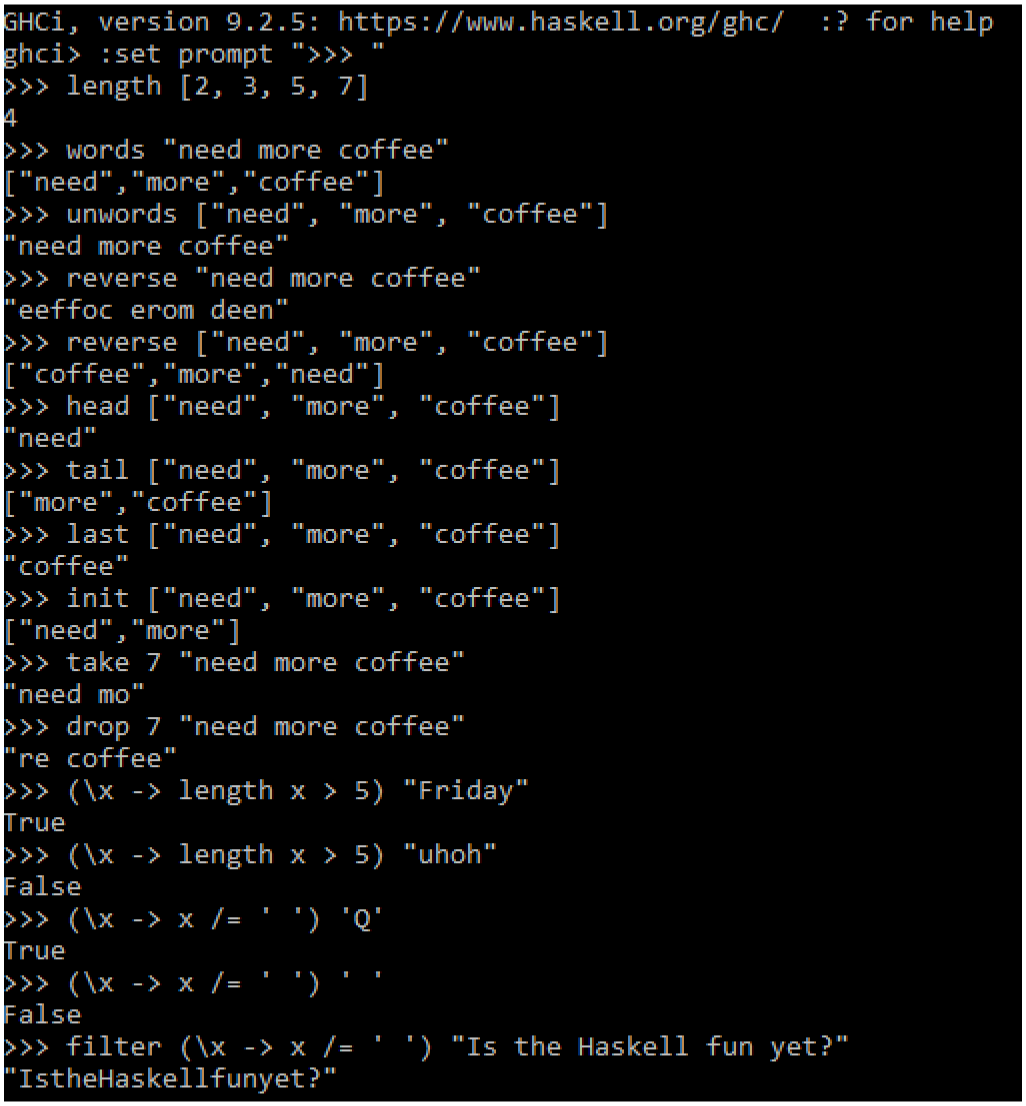
Haskell Programming Assignment: Various Computions

Abstract

The point of this assignment is to see how to use the functionality of Haskell and be able to play around with the language as well as see the fascination behind it.

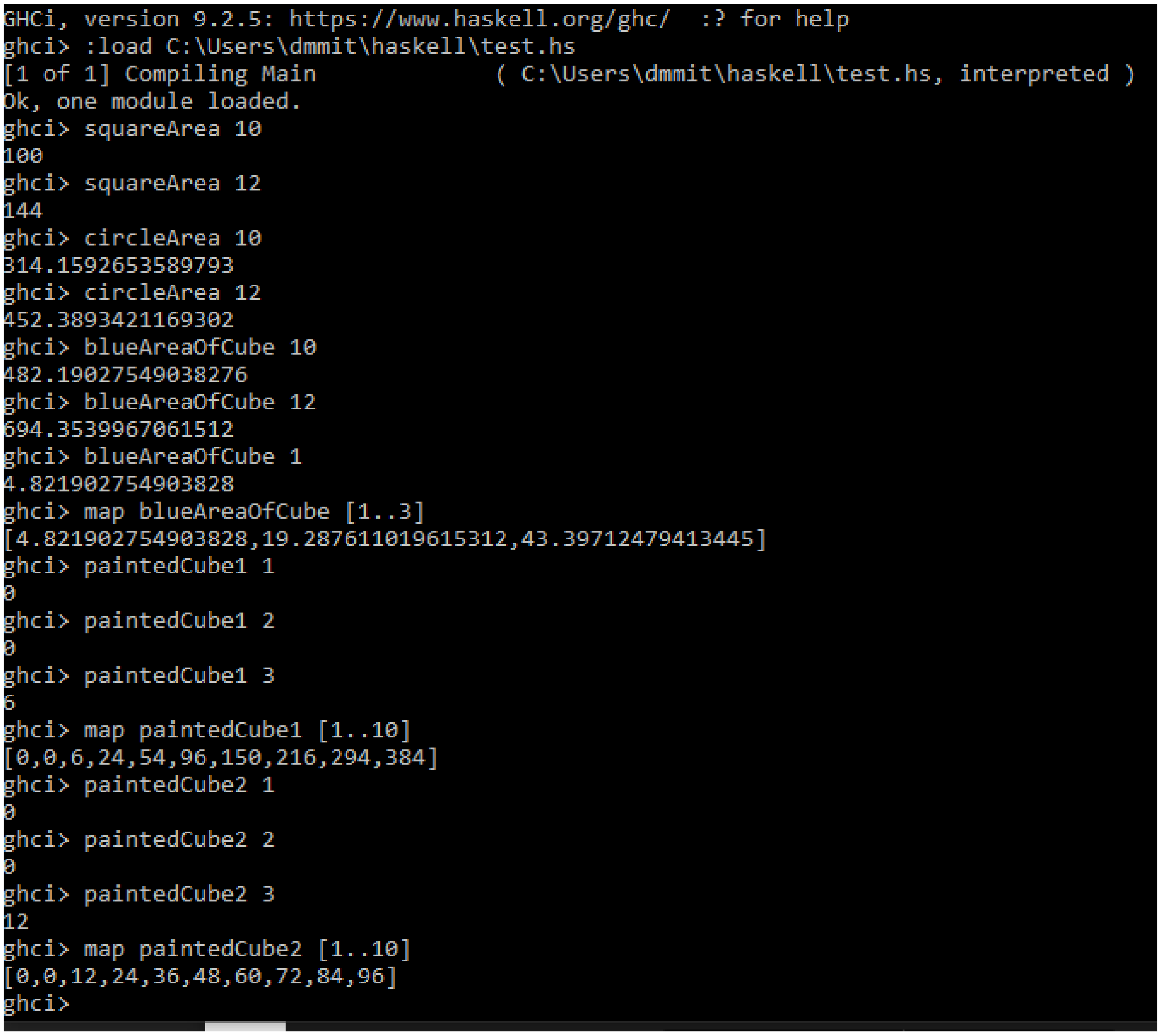
Task 1 - Mindfully Mimicking the Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
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 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?"
```

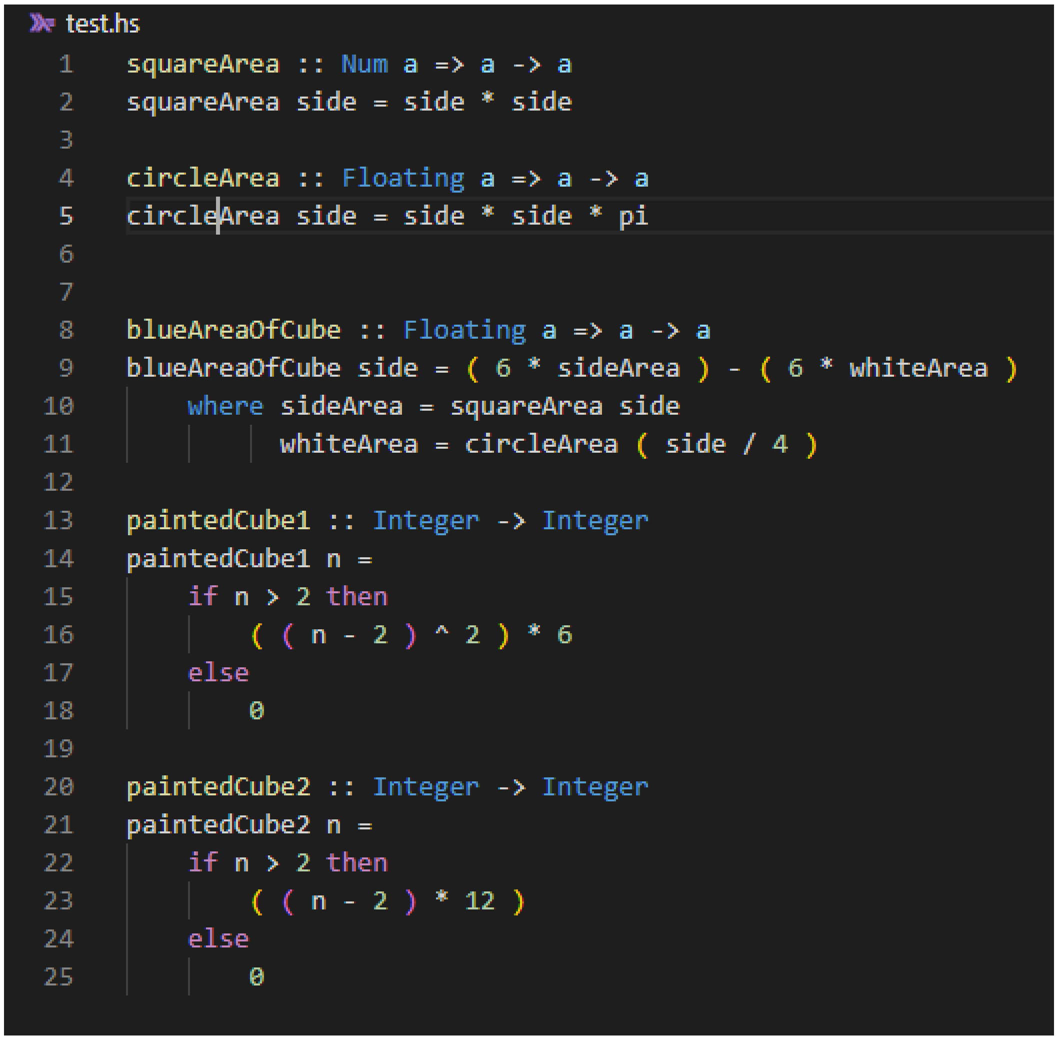
Task 2 - Numeric Function Definitions

Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test.hs, interpreted )
Ok, one module loaded.
ghci> squareArea 10
100
ghci> squareArea 12
144
ghci> circleArea 10
314.1592653589793
ghci> circleArea 12
452.3893421169302
ghci> blueAreaOfCube 10
482.19027549038276
ghci> blueAreaOfCube 12
694.3539967061512
ghci> blueAreaOfCube 1
4.821902754903828
ghci> map blueAreaOfCube [1..3]
[4.821902754903828,19.287611019615312,43.39712479413445]
ghci> paintedCube1 1
0
ghci> paintedCube1 2
0
ghci> paintedCube1 3
6
ghci> map paintedCube1 [1..10]
[0,0,6,24,54,96,150,216,294,384]
ghci> paintedCube2 1
0
ghci> paintedCube2 2
0
ghci> paintedCube2 3
12
ghci> map paintedCube2 [1..10]
[0,0,12,24,36,48,60,72,84,96]
ghci>
```

Code

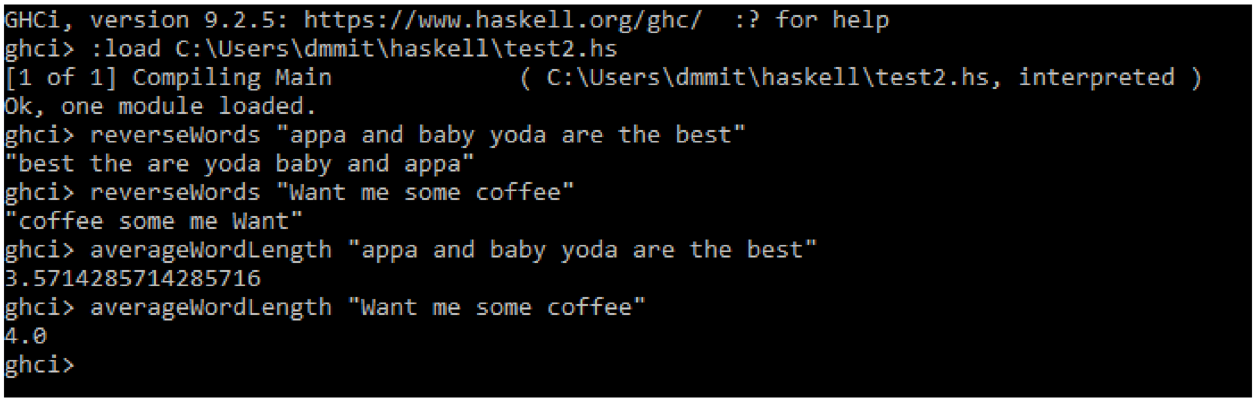


test.hs

```haskell
squareArea :: Num a => a -> a
squareArea side = side * side

circleArea :: Floating a => a -> a
circleArea side = side * side * pi


blueAreaOfCube :: Floating a => a -> a
blueAreaOfCube side = ( 6 * sideArea ) - ( 6 * whiteArea )
    where sideArea = squareArea side
          whiteArea = circleArea ( side / 4 )

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

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

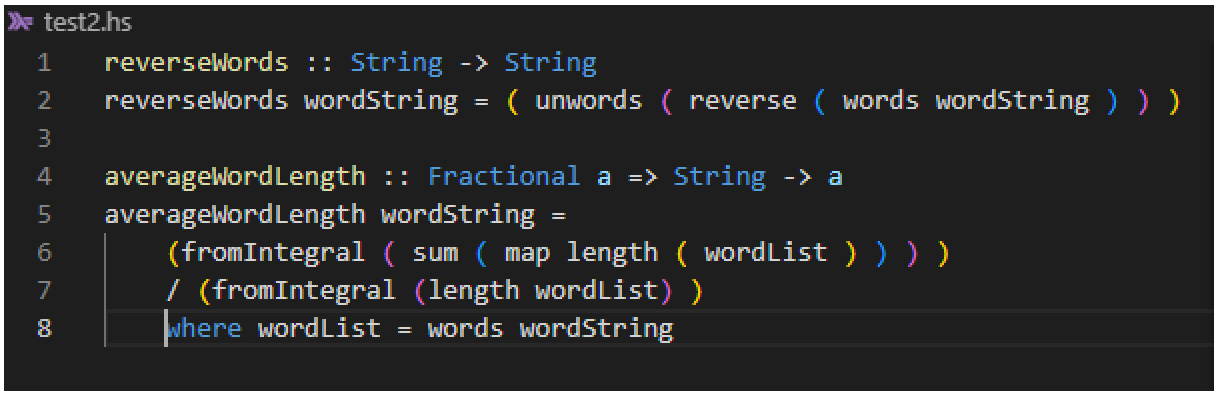
Task 3 - Puzzlers

Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test2.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test2.hs, interpreted )
Ok, one module loaded.
ghci> reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
ghci> reverseWords "Want me some coffee"
"coffee some me Want"
ghci> averageWordLength "appa and baby yoda are the best"
3.5714285714285716
ghci> averageWordLength "Want me some coffee"
4.0
ghci>
```

Code

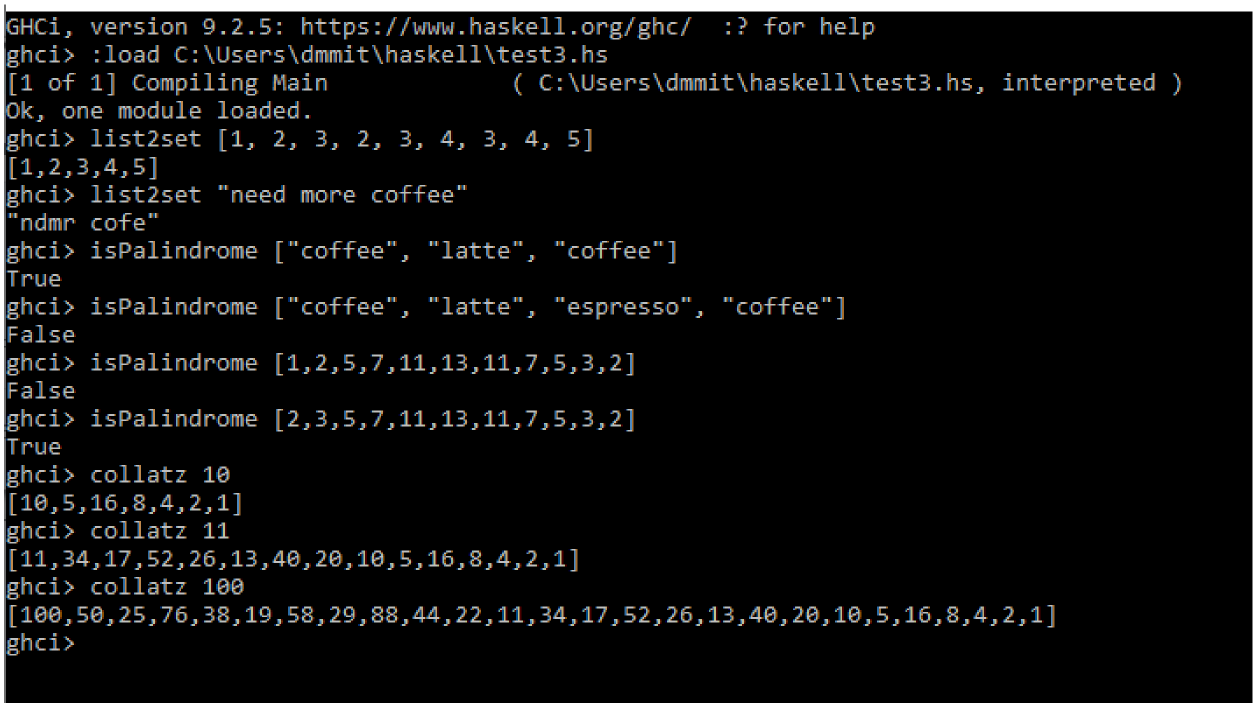


test2.hs

```haskell
reverseWords :: String -> String
reverseWords wordString = ( unwords ( reverse ( words wordString ) ) )

averageWordLength :: Fractional a => String -> a
averageWordLength wordString =
    (fromIntegral ( sum ( map length ( wordList ) ) ) )
    / (fromIntegral (length wordList) )
    where wordList = words wordString
```

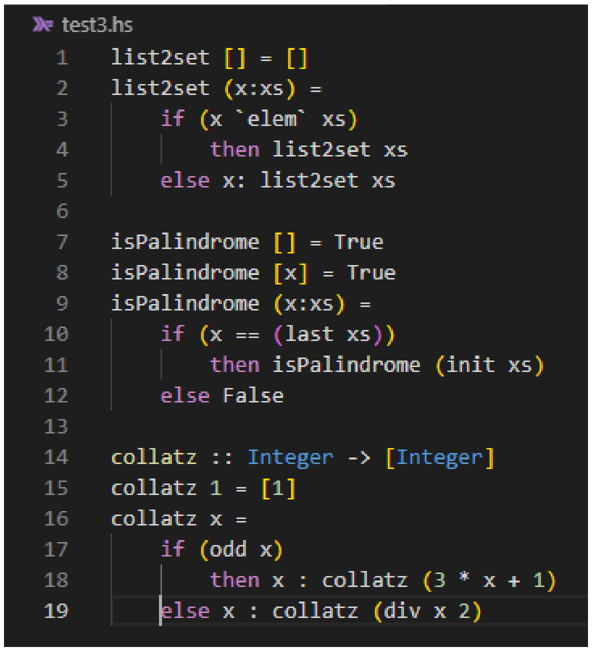
Task 4 - Recursive List Processors

Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test3.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test3.hs, interpreted )
Ok, one module loaded.
ghci> list2set [1, 2, 3, 2, 3, 4, 3, 4, 5]
[1,2,3,4,5]
ghci> list2set "need more coffee"
"ndmr cofe"
ghci> isPalindrome ["coffee", "latte", "coffee"]
True
ghci> isPalindrome ["coffee", "latte", "espresso", "coffee"]
False
ghci> isPalindrome [1,2,5,7,11,13,11,7,5,3,2]
False
ghci> isPalindrome [2,3,5,7,11,13,11,7,5,3,2]
True
ghci> collatz 10
[10,5,16,8,4,2,1]
ghci> collatz 11
[11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
ghci> collatz 100
[100,50,25,76,38,19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
ghci>
```

Code

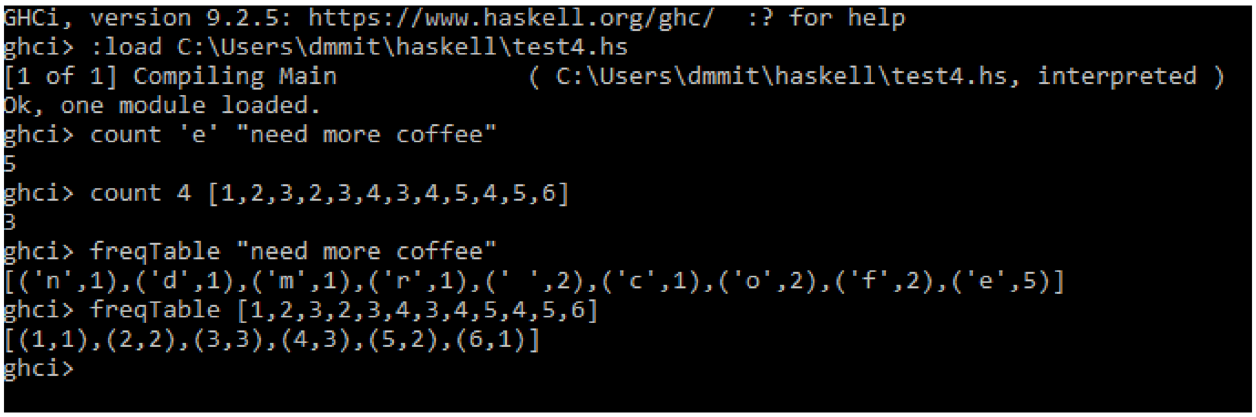


test3.hs

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

isPalindrome [] = True
isPalindrome [x] = True
isPalindrome (x:xs) =
    if (x == (last xs))
        then isPalindrome (init xs)
    else False

collatz :: Integer -> [Integer]
collatz 1 = [1]
collatz x =
    if (odd x)
        then x : collatz (3 * x + 1)
    else x : collatz (div x 2)
```

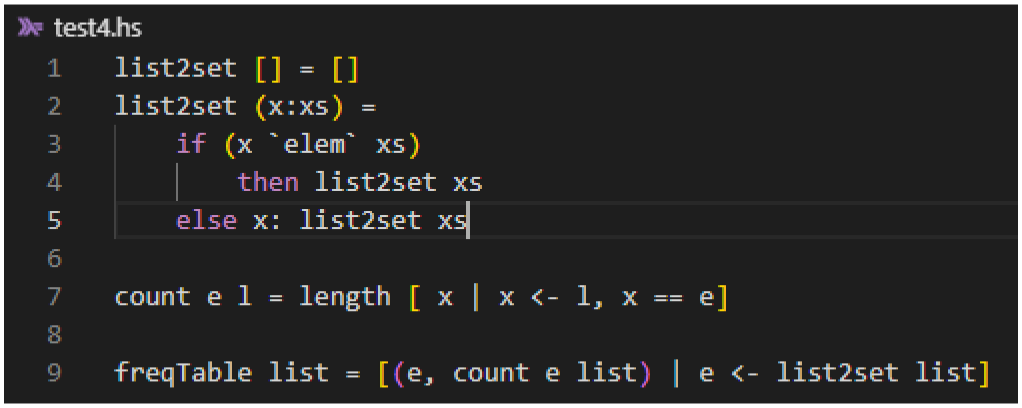
Task 5 - List Comprehensions

Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test4.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test4.hs, interpreted )
Ok, one module loaded.
ghci> count 'e' "need more coffee"
5
ghci> count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
3
ghci> freqTable "need more coffee"
[('n',1),('d',1),('m',1),('r',1),(' ',2),('c',1),('o',2),('f',2),('e',5)]
ghci> freqTable [1,2,3,2,3,4,3,4,5,4,5,6]
[(1,1),(2,2),(3,3),(4,3),(5,2),(6,1)]
ghci>
```

Code

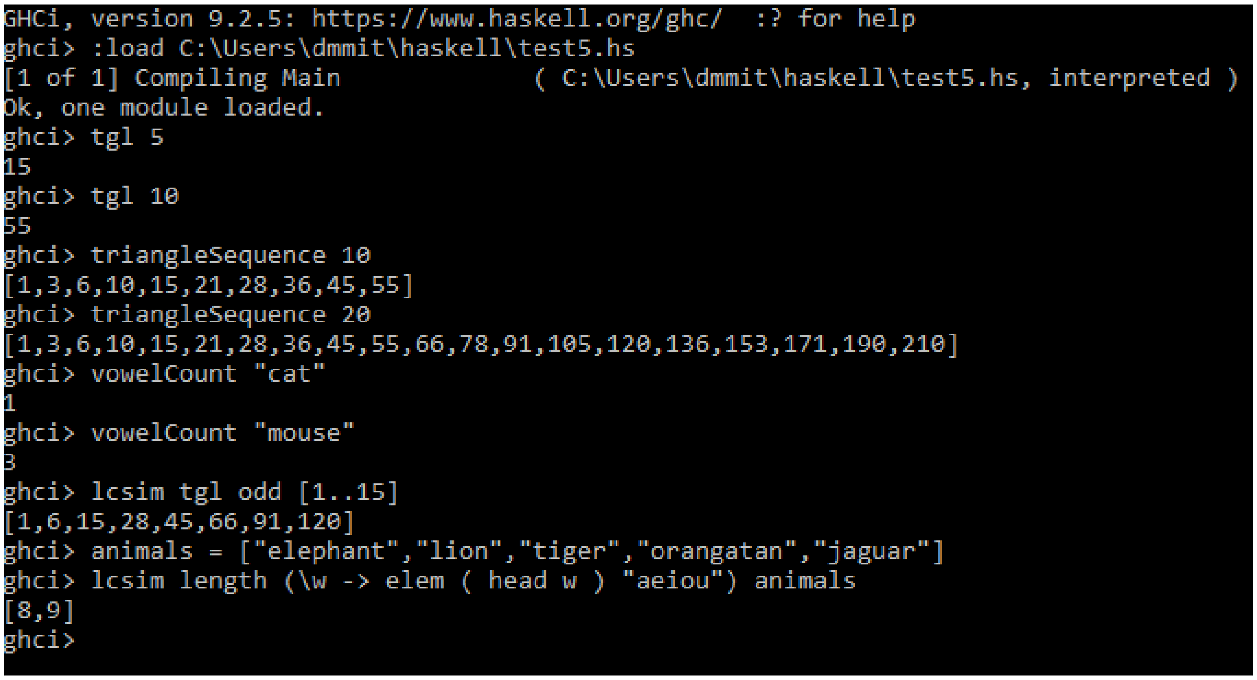


test4.hs

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

count e l = length [ x | x <- l, x == e]

freqTable list = [(e, count e list) | e <- list2set list]
```

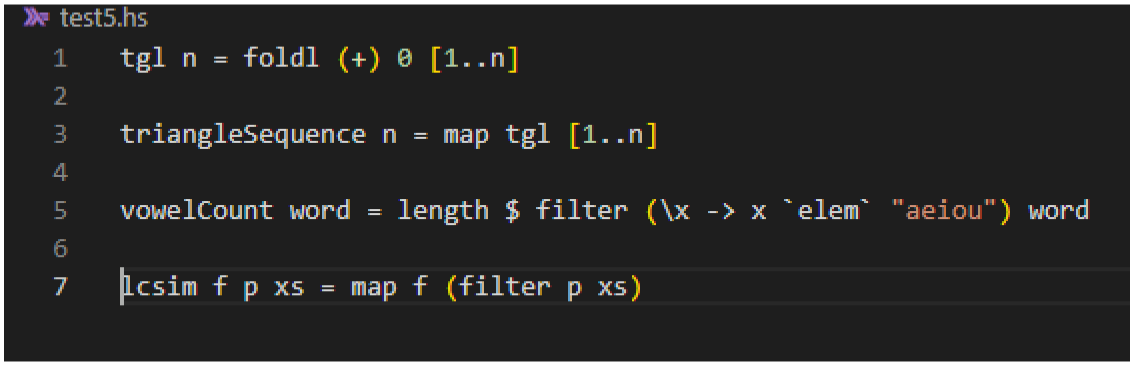
Task 6 - Higher Order Functions

Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test5.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test5.hs, interpreted )
Ok, one module loaded.
ghci> tgl 5
15
ghci> tgl 10
55
ghci> triangleSequence 10
[1,3,6,10,15,21,28,36,45,55]
ghci> triangleSequence 20
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210]
ghci> vowelCount "cat"
1
ghci> vowelCount "mouse"
3
ghci> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
ghci> animals = ["elephant","lion","tiger","orangatan","jaguar"]
ghci> lcsim length (\w -> elem ( head w ) "aeiou") animals
[8,9]
ghci>
```

Code

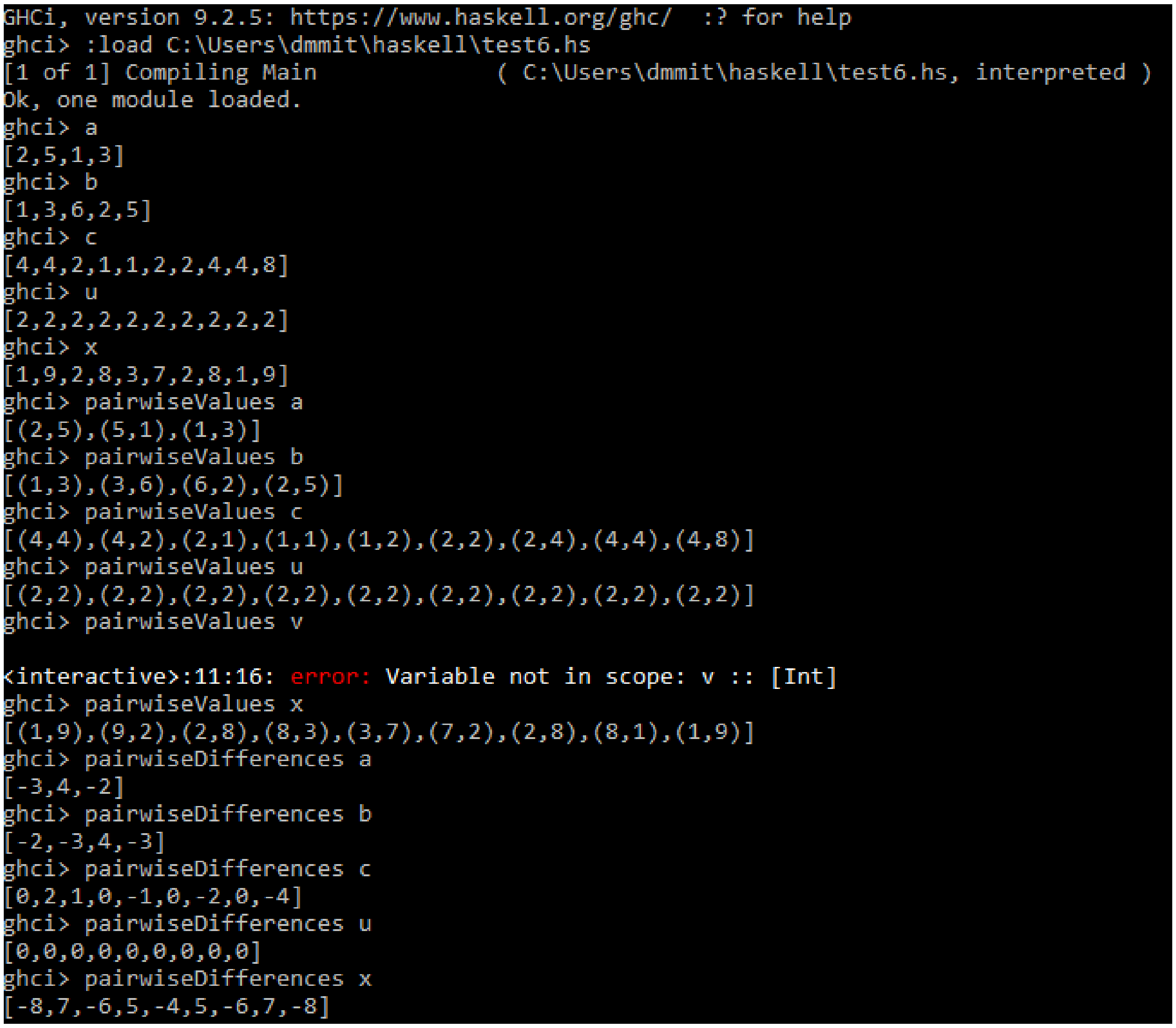


test5.hs

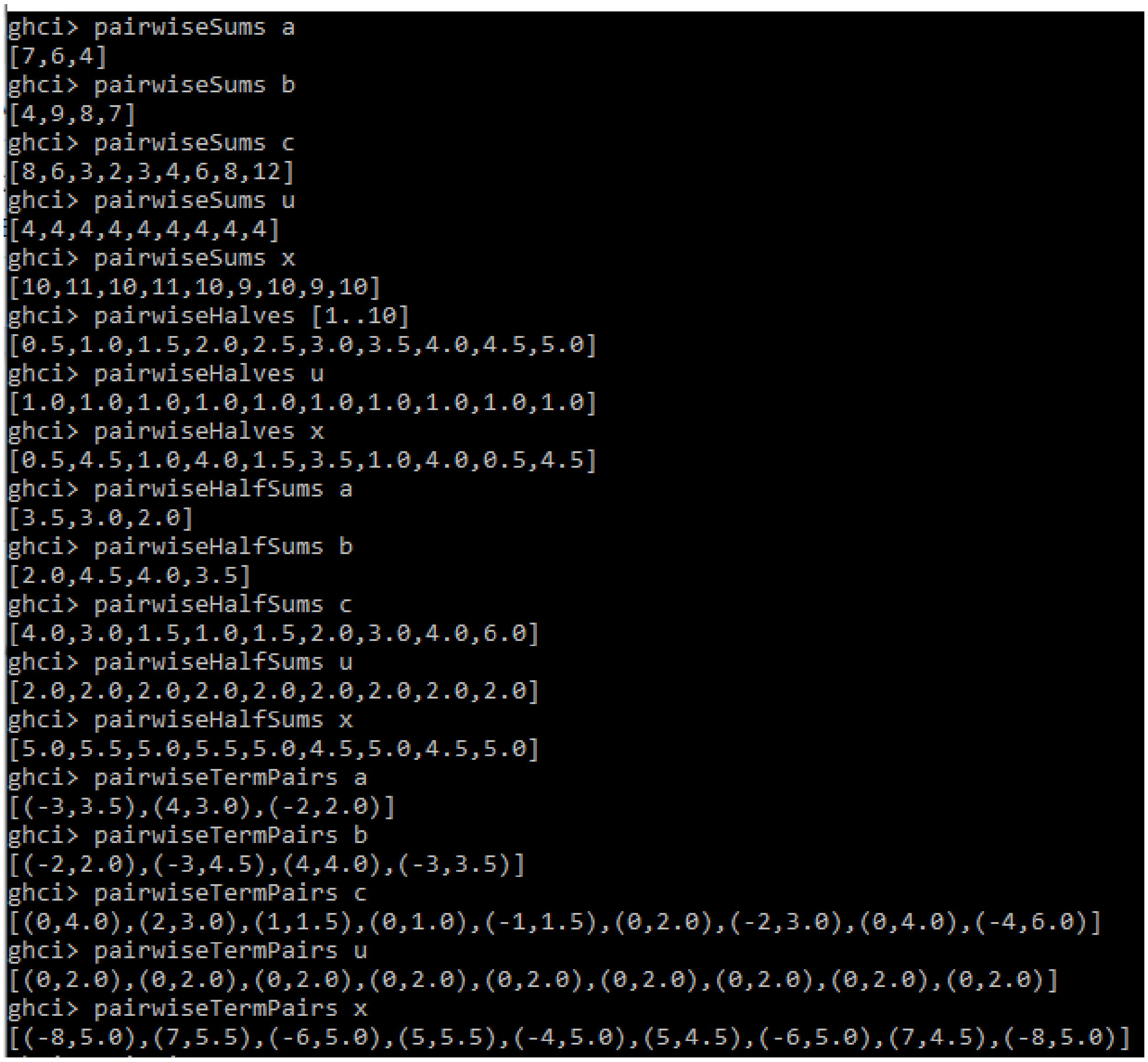
```haskell
tgl n = foldl (+) 0 [1..n]

triangleSequence n = map tgl [1..n]

vowelCount word = length $ filter (\x -> x `elem` "aeiou") word

lcsim f p xs = map f (filter p xs)
```

Task 7 - An Interesting Statistic: nPVI

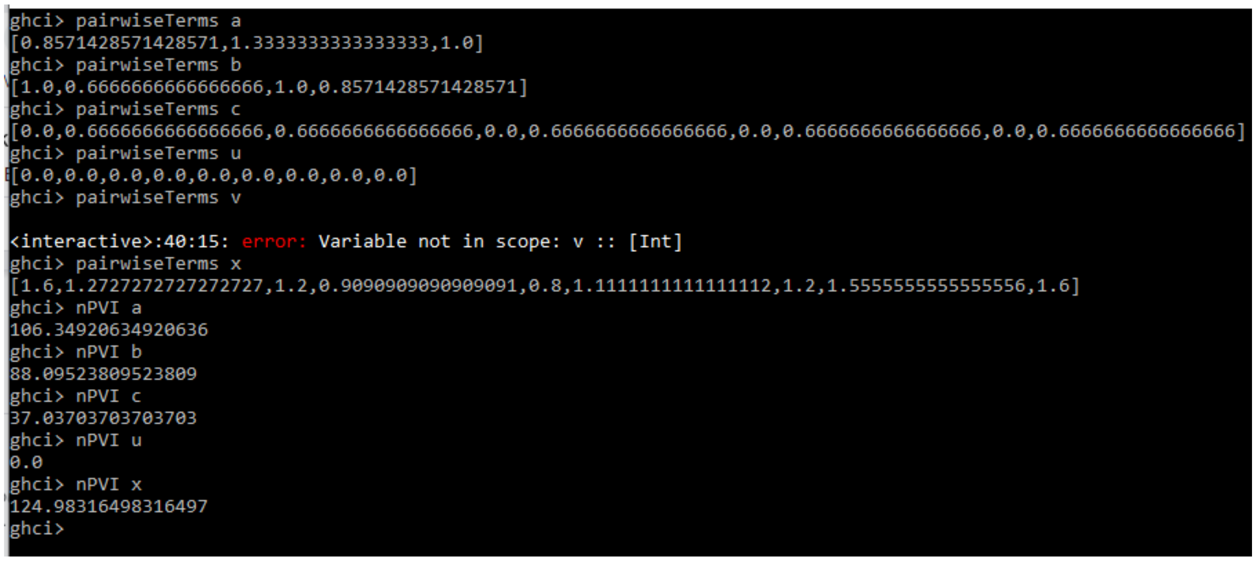
Demo



```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test6.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test6.hs, interpreted )
Ok, one module loaded.
ghci> a
[2,5,1,3]
ghci> b
[1,3,6,2,5]
ghci> c
[4,4,2,1,1,2,2,4,4,8]
ghci> u
[2,2,2,2,2,2,2,2,2,2]
ghci> x
[1,9,2,8,3,7,2,8,1,9]
ghci> pairwiseValues a
[(2,5),(5,1),(1,3)]
ghci> pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
ghci> pairwiseValues c
[(4,4),(4,2),(2,1),(1,1),(1,2),(2,2),(2,4),(4,4),(4,8)]
ghci> pairwiseValues u
[(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2)]
ghci> pairwiseValues v

<interactive>:11:16: error: Variable not in scope: v :: [Int]
ghci> pairwiseValues x
[(1,9),(9,2),(2,8),(8,3),(3,7),(7,2),(2,8),(8,1),(1,9)]
ghci> pairwiseDifferences a
[-3,4,-2]
ghci> pairwiseDifferences b
[-2,-3,4,-3]
ghci> pairwiseDifferences c
[0,2,1,0,-1,0,-2,0,-4]
ghci> pairwiseDifferences u
[0,0,0,0,0,0,0,0,0]
ghci> pairwiseDifferences x
[-8,7,-6,5,-4,5,-6,7,-8]
```

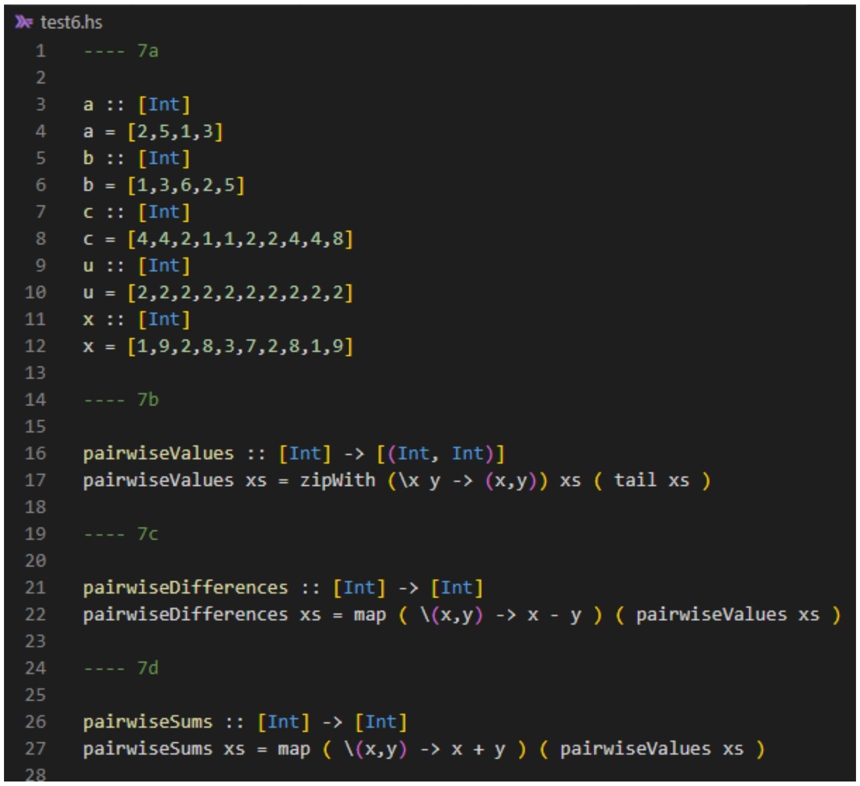


```
ghci> pairwiseSums a
[7,6,4]
ghci> pairwiseSums b
[4,9,8,7]
ghci> pairwiseSums c
[8,6,3,2,3,4,6,8,12]
ghci> pairwiseSums u
[4,4,4,4,4,4,4,4,4]
ghci> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
ghci> pairwiseHalves [1..10]
[0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
ghci> pairwiseHalves u
[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
ghci> pairwiseHalves x
[0.5,4.5,1.0,4.0,1.5,3.5,1.0,4.0,0.5,4.5]
ghci> pairwiseHalfSums a
[3.5,3.0,2.0]
ghci> pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
ghci> pairwiseHalfSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,6.0]
ghci> pairwiseHalfSums u
[2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0]
ghci> pairwiseHalfSums x
[5.0,5.5,5.0,5.5,5.0,4.5,5.0,4.5,5.0]
ghci> pairwiseTermPairs a
[(-3,3.5),(4,3.0),(-2,2.0)]
ghci> pairwiseTermPairs b
[(-2,2.0),(-3,4.5),(4,4.0),(-3,3.5)]
ghci> pairwiseTermPairs c
[(0,4.0),(2,3.0),(1,1.5),(0,1.0),(-1,1.5),(0,2.0),(-2,3.0),(0,4.0),(-4,6.0)]
ghci> pairwiseTermPairs u
[(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0)]
ghci> pairwiseTermPairs x
[(-8,5.0),(7,5.5),(-6,5.0),(5,5.5),(-4,5.0),(5,4.5),(-6,5.0),(7,4.5),(-8,5.0)]
```

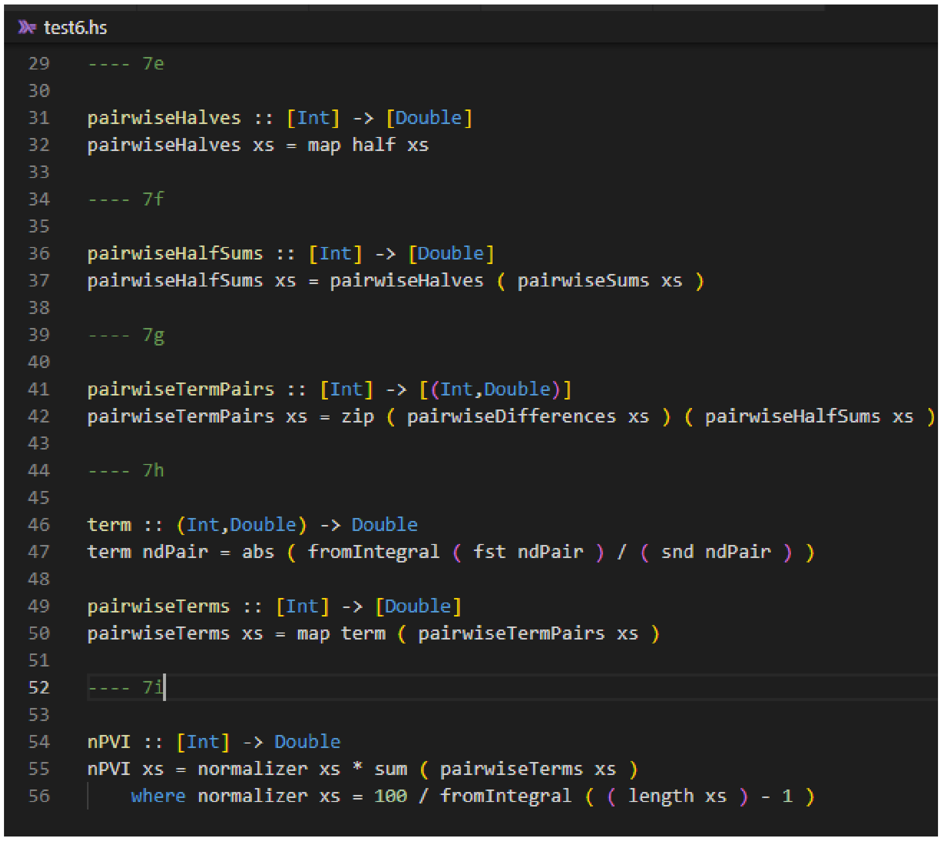


```
ghci> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
ghci> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
ghci> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666]
ghci> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
ghci> pairwiseTerms v

<interactive>:40:15: error: Variable not in scope: v :: [Int]
ghci> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555555555555556,1.6]
ghci> nPVI a
106.34920634920636
ghci> nPVI b
88.09523809523809
ghci> nPVI c
37.03703703703703
ghci> nPVI u
0.0
ghci> nPVI x
124.98316498316497
ghci>
```

Code



```haskell
test6.hs
---- 7a

a :: [Int]
a = [2,5,1,3]
b :: [Int]
b = [1,3,6,2,5]
c :: [Int]
c = [4,4,2,1,1,2,2,4,4,8]
u :: [Int]
u = [2,2,2,2,2,2,2,2,2,2]
x :: [Int]
x = [1,9,2,8,3,7,2,8,1,9]

---- 7b

pairwiseValues :: [Int] -> [(Int, Int)]
pairwiseValues xs = zipWith (\x y -> (x,y)) xs ( tail xs )

---- 7c

pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences xs = map ( \(x,y) -> x - y ) ( pairwiseValues xs )

---- 7d

pairwiseSums :: [Int] -> [Int]
pairwiseSums xs = map ( \(x,y) -> x + y ) ( pairwiseValues xs )
```

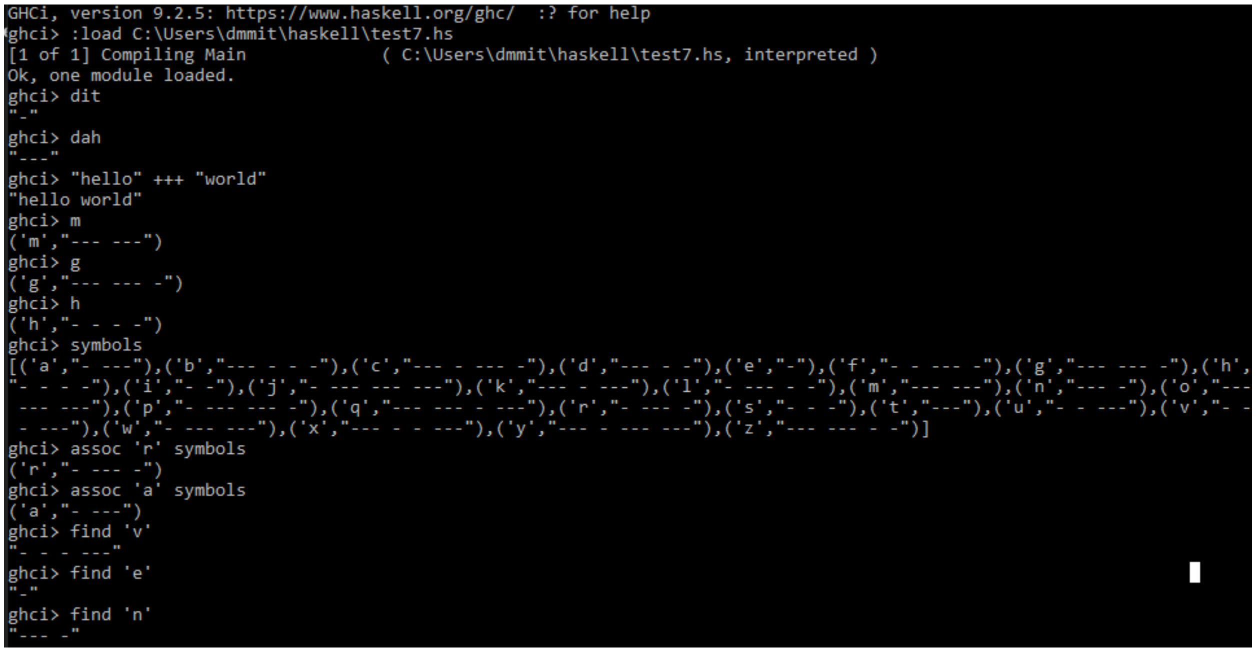


test6.hs

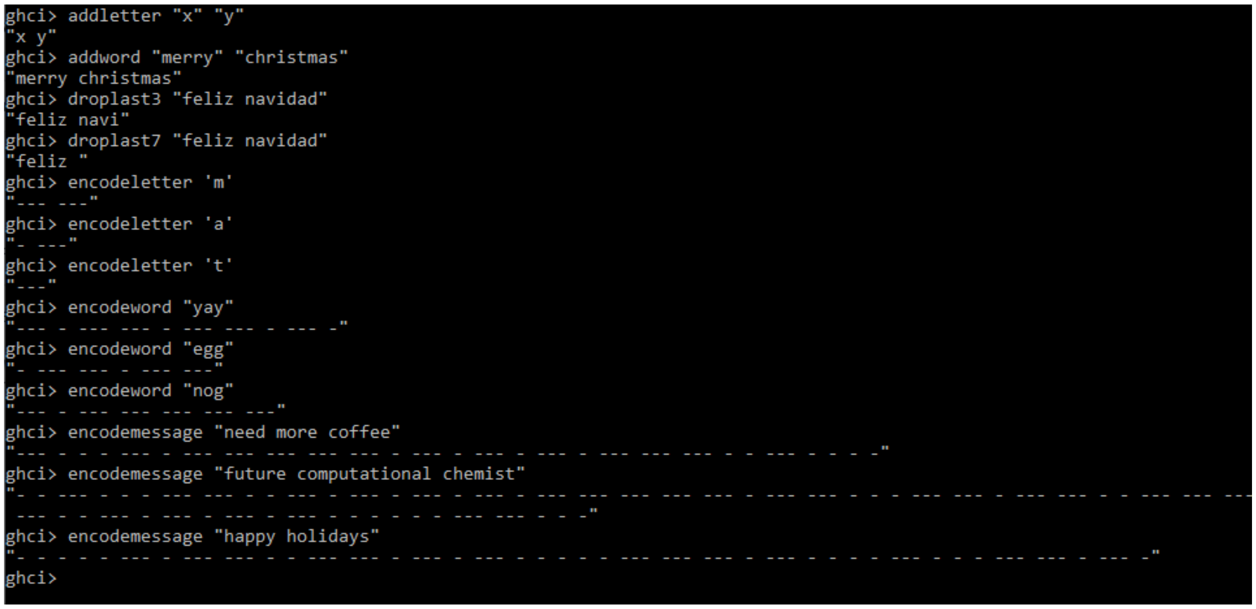
```haskell
---- 7e

pairwiseHalves :: [Int] -> [Double]
pairwiseHalves xs = map half xs

---- 7f

pairwiseHalfSums :: [Int] -> [Double]
pairwiseHalfSums xs = pairwiseHalves ( pairwiseSums xs )

---- 7g

pairwiseTermPairs :: [Int] -> [(Int,Double)]
pairwiseTermPairs xs = zip ( pairwiseDifferences xs ) ( pairwiseHalfSums xs )

---- 7h

term :: (Int,Double) -> Double
term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )

pairwiseTerms :: [Int] -> [Double]
pairwiseTerms xs = map term ( pairwiseTermPairs xs )

---- 7i

nPVI :: [Int] -> Double
nPVI xs = normalizer xs * sum ( pairwiseTerms xs )
    where normalizer xs = 100 / fromIntegral ( ( length xs ) - 1 )
```

Task 8 - Historic Code: The Dit Dah Code

Demo

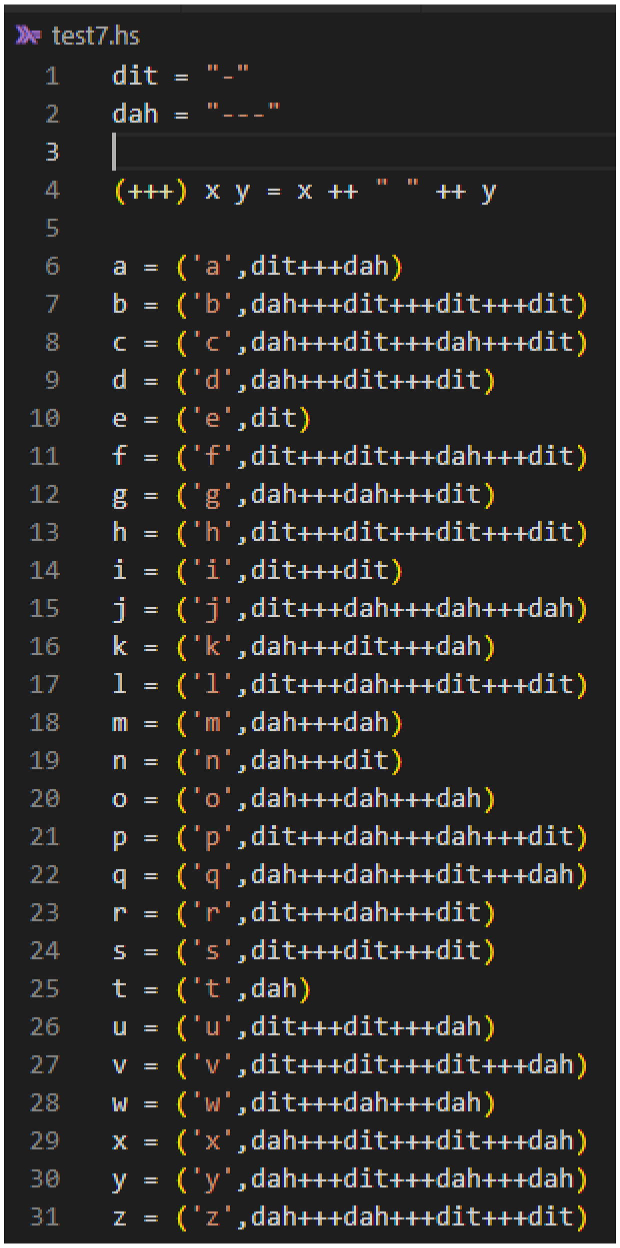


```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for help
ghci> :load C:\Users\dmmit\haskell\test7.hs
[1 of 1] Compiling Main             ( C:\Users\dmmit\haskell\test7.hs, interpreted )
Ok, one module loaded.
ghci> dit
"-"
ghci> dah
"---"
ghci> "hello" +++ "world"
"hello world"
ghci> m
('m',"--- ---")
ghci> g
('g',"--- --- -")
ghci> h
('h',"- - - -")
ghci> symbols
[('a',"- ---"),('b',"--- - - -"),('c',"--- - --- -"),('d',"--- - -"),('e',"-"),('f',"- - --- -"),('g',"--- --- -"),('h',
"- - - -"),('i',"- -"),('j',"- --- --- ---"),('k',"--- - ---"),('l',"- --- - -"),('m',"--- ---"),('n',"--- -"),('o',"---
 --- ---"),('p',"- --- --- -"),('q',"--- --- - ---"),('r',"- --- -"),('s',"- - -"),('t',"---"),('u',"- - ---"),('v',"- -
 - ---"),('w',"- --- ---"),('x',"--- - - ---"),('y',"--- - --- ---"),('z',"--- --- - -")]
ghci> assoc 'r' symbols
('r',"- --- -")
ghci> assoc 'a' symbols
('a',"- ---")
ghci> find 'v'
"- - - ---"
ghci> find 'e'
"-"
ghci> find 'n'
"--- -"
```



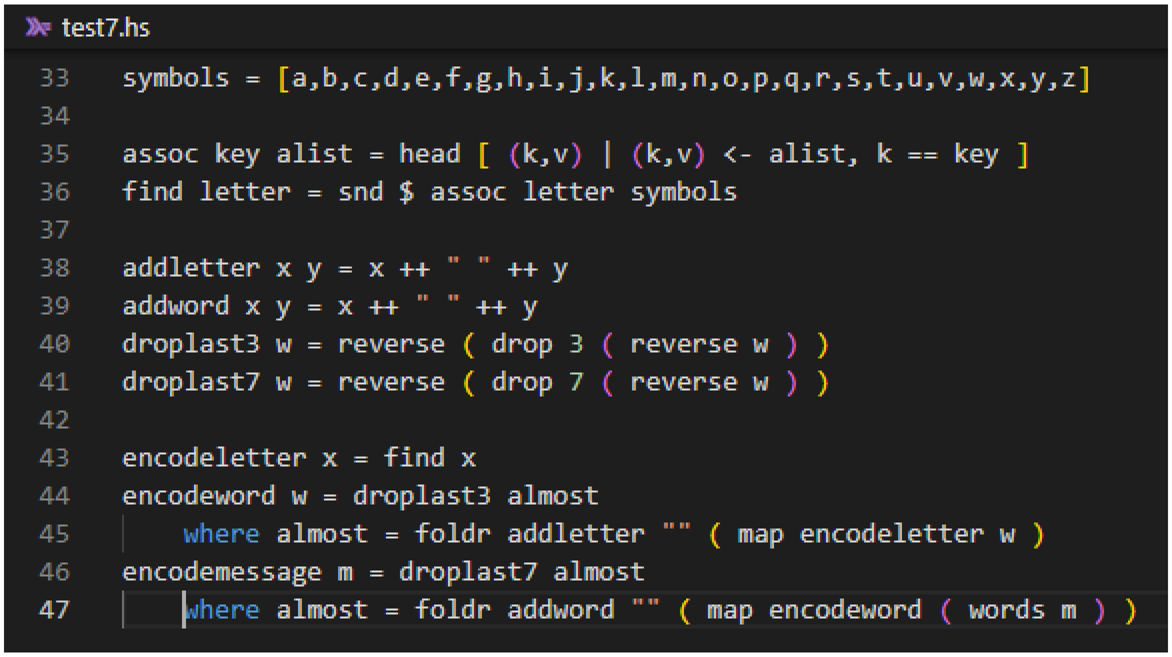
```
ghci> addletter "x" "y"
"x y"
ghci> addword "merry" "christmas"
"merry christmas"
ghci> droplast3 "feliz navidad"
"feliz navi"
ghci> droplast7 "feliz navidad"
"feliz "
ghci> encodeletter 'm'
"--- ---"
ghci> encodeletter 'a'
"- ---"
ghci> encodeletter 't'
"---"
ghci> encodeword "yay"
"--- - --- --- - --- --- - --- -"
ghci> encodeword "egg"
"- --- --- - --- ---"
ghci> encodeword "nog"
"--- - --- --- --- --- ---"
ghci> encodemessage "need more coffee"
"--- - - - --- - --- --- --- --- --- - --- - --- - --- - --- --- --- - - --- - - - -"
ghci> encodemessage "future computational chemist"
"- - --- - - - --- --- - - --- - --- - --- - --- - --- --- --- --- --- - --- --- - - - --- --- - --- --- - - --- --- ---
 --- - - --- - --- - --- - --- - - - - - - --- --- - - -"
ghci> encodemessage "happy holidays"
"- - - - - - --- - --- --- - - --- --- - --- - --- - - - - - --- --- --- - --- - - - - --- - - - --- --- - --- -"
ghci>
```

Code



test7.hs

```haskell
dit = "-"
dah = "---"

(+++) x y = x ++ " " ++ y

a = ('a',dit+++dah)
b = ('b',dah+++dit+++dit+++dit)
c = ('c',dah+++dit+++dah+++dit)
d = ('d',dah+++dit+++dit)
e = ('e',dit)
f = ('f',dit+++dit+++dah+++dit)
g = ('g',dah+++dah+++dit)
h = ('h',dit+++dit+++dit+++dit)
i = ('i',dit+++dit)
j = ('j',dit+++dah+++dah+++dah)
k = ('k',dah+++dit+++dah)
l = ('l',dit+++dah+++dit+++dit)
m = ('m',dah+++dah)
n = ('n',dah+++dit)
o = ('o',dah+++dah+++dah)
p = ('p',dit+++dah+++dah+++dit)
q = ('q',dah+++dah+++dit+++dah)
r = ('r',dit+++dah+++dit)
s = ('s',dit+++dit+++dit)
t = ('t',dah)
u = ('u',dit+++dit+++dah)
v = ('v',dit+++dit+++dit+++dah)
w = ('w',dit+++dah+++dah)
x = ('x',dah+++dit+++dit+++dah)
y = ('y',dah+++dit+++dah+++dah)
z = ('z',dah+++dah+++dit+++dit)
```


test7.hs

```haskell
symbols = [a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z]

assoc key alist = head [ (k,v) | (k,v) <- alist, k == key ]
find letter = snd $ assoc letter symbols

addletter x y = x ++ " " ++ y
addword x y = x ++ "  " ++ y
droplast3 w = reverse ( drop 3 ( reverse w ) )
droplast7 w = reverse ( drop 7 ( reverse w ) )

encodeletter x = find x
encodeword w = droplast3 almost
    where almost = foldr addletter "" ( map encodeletter w )
encodemessage m = droplast7 almost
    where almost = foldr addword "" ( map encodeword ( words m ) )
```