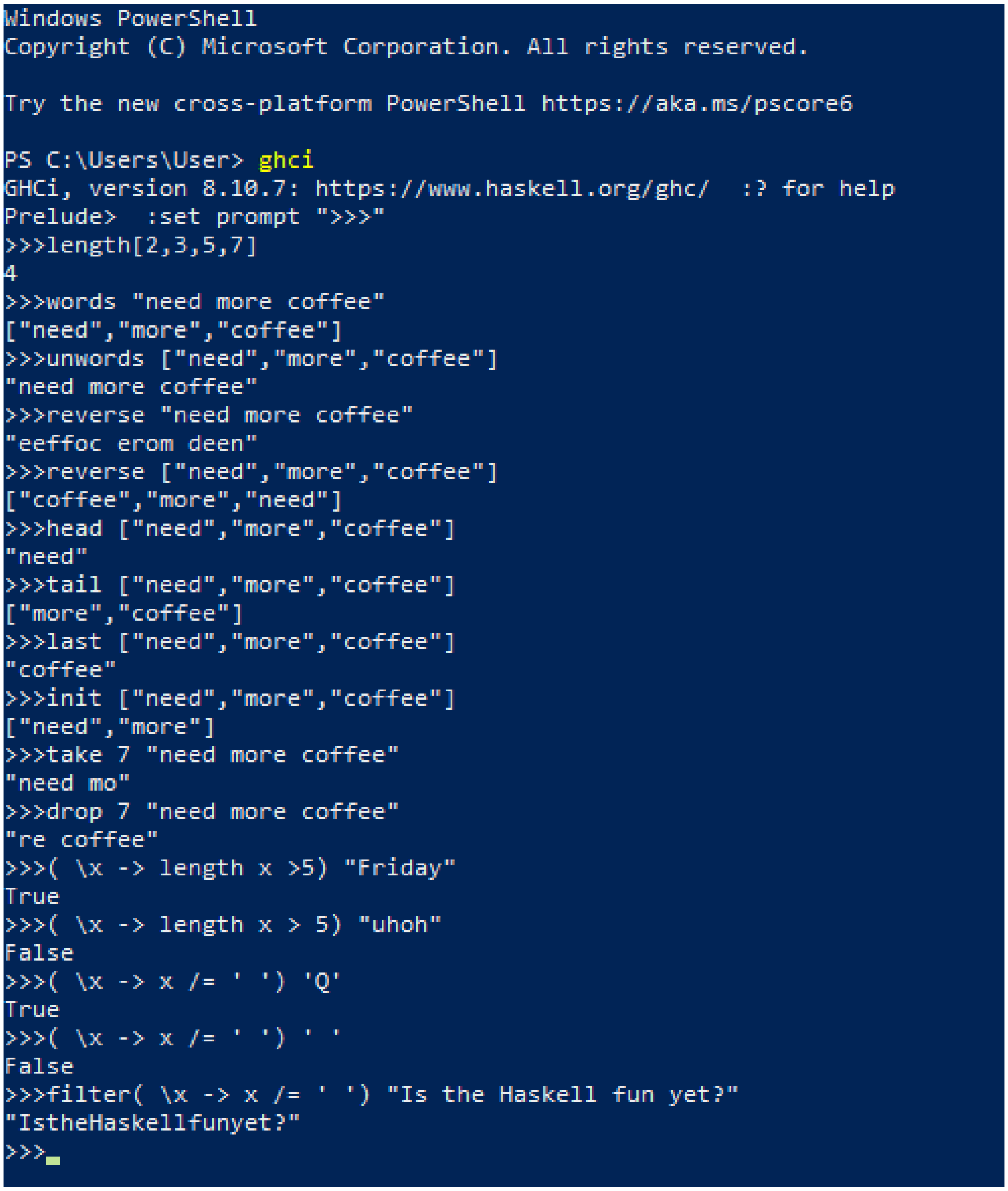
Haskell Programming Assignment

Abstract

The purpose of this assignment is to demonstrate the functionality of the Haskell programming language. The assignment allows us to test values and expressions, manipulate lists in different ways, and develop higher order functions to manipulate data.

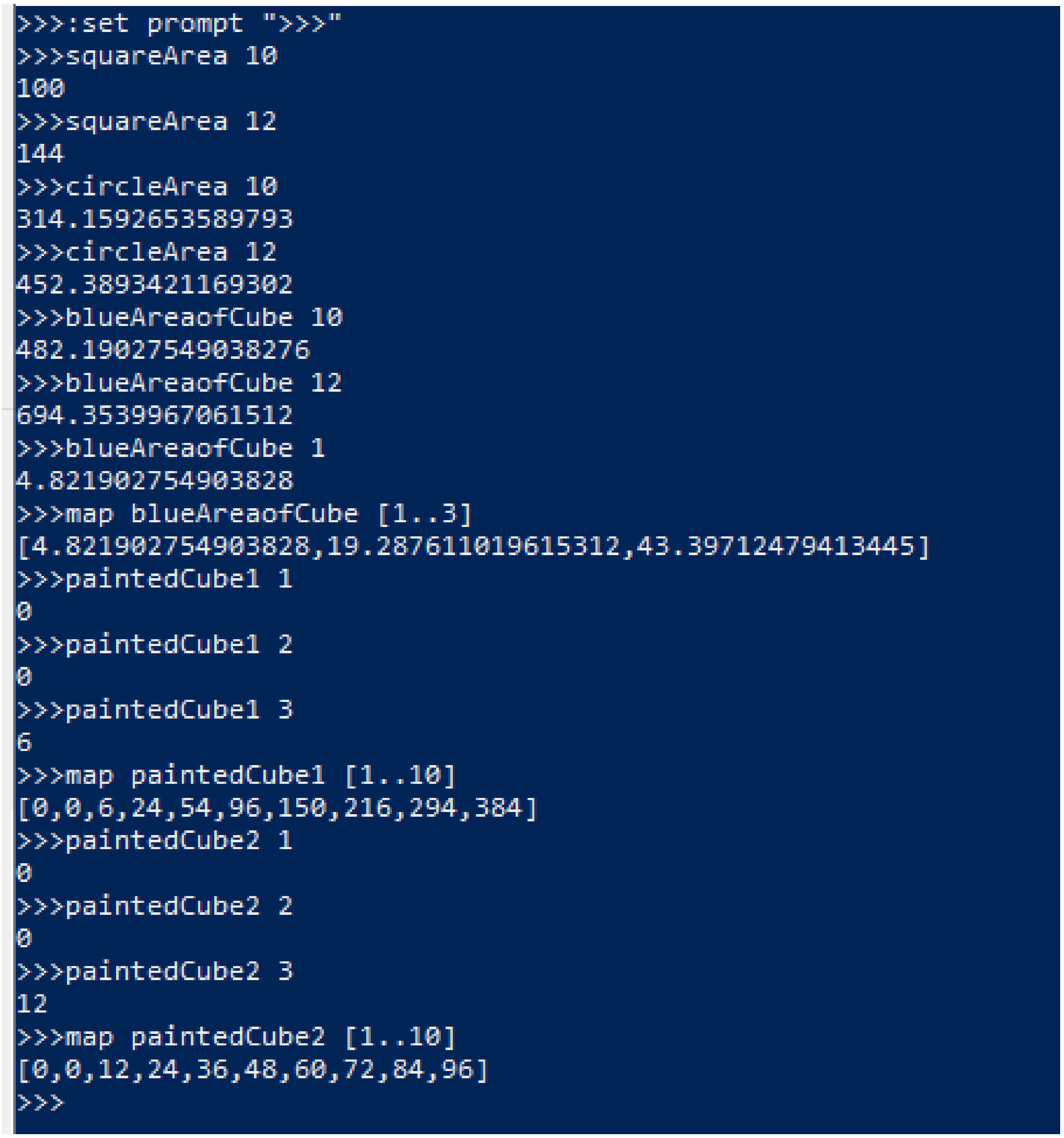
Task 1 – Mindfully Mimicking the Demo



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\User> ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude>  :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

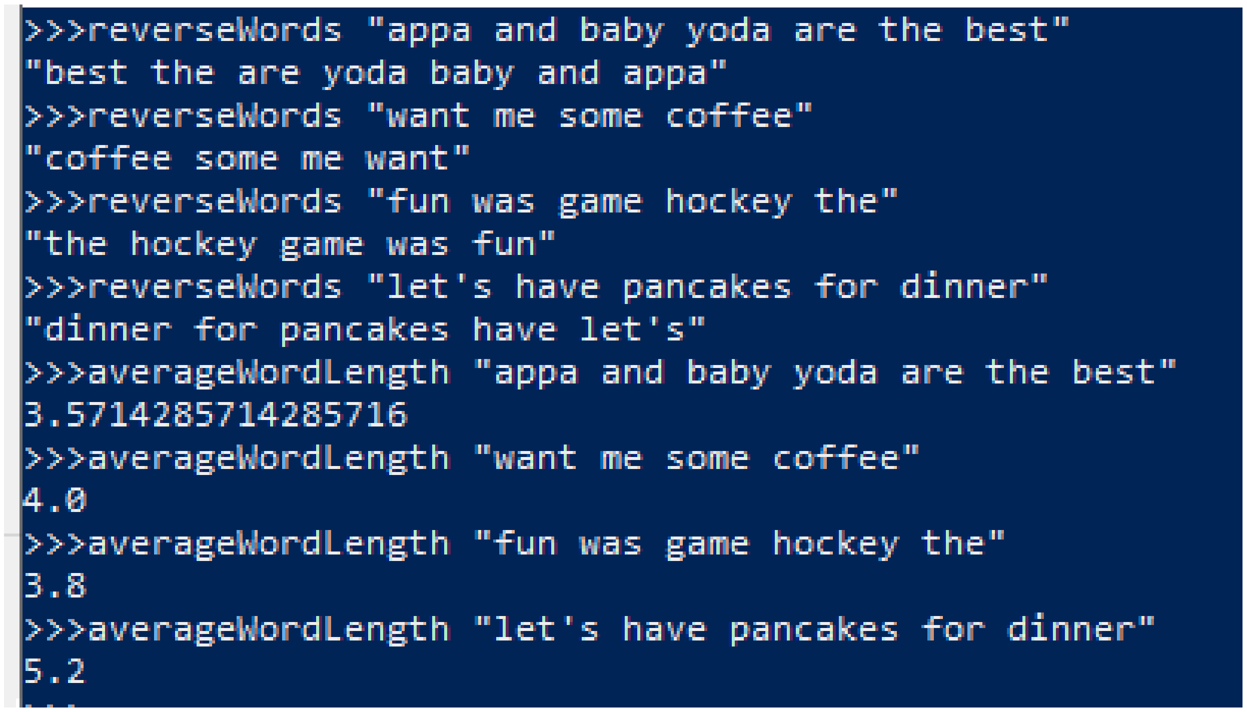
```
--------------------------------------------------------------------------
--- csc344_haskell.hs contains some functions for the haskell assignment
--------------------------------------------------------------------------
-- squareArea

squareArea :: Int -> Int
squareArea x =(x * x)

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

circleArea :: Double -> Double
circleArea r = (pi *(r * r))
--------------------------------------------------------------------------
-- blueAreaofCube

blueAreaofCube :: Double -> Double
blueAreaofCube x = (x * x * 6) - (pi * ((1/4 * x) *(1/4 * x) * 6))

--------------------------------------------------------------------------
-- paintedCube1

paintedCube1 :: Int -> Int
paintedCube1 x = if (x > 2) then (6 *(x - 2)^2) else 0

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

paintedCube2 :: Int -> Int
paintedCube2 x = if (x > 2) then (12 * (x - 2)) else 0
--------------------------------------------------------------------------
```



```
>>>:set prompt ">>>"
>>>squareArea 10
100
>>>squareArea 12
144
>>>circleArea 10
314.1592653589793
>>>circleArea 12
452.3893421169302
>>>blueAreaofCube 10
482.19027549038276
>>>blueAreaofCube 12
694.3539967061512
>>>blueAreaofCube 1
4.821902754903828
>>>map blueAreaofCube [1..3]
[4.821902754903828,19.287611019615312,43.39712479413445]
>>>paintedCube1 1
0
>>>paintedCube1 2
0
>>>paintedCube1 3
6
>>>map paintedCube1 [1..10]
[0,0,6,24,54,96,150,216,294,384]
>>>paintedCube2 1
0
>>>paintedCube2 2
0
>>>paintedCube2 3
12
>>>map paintedCube2 [1..10]
[0,0,12,24,36,48,60,72,84,96]
>>>
```

Task 3 – Puzzlers

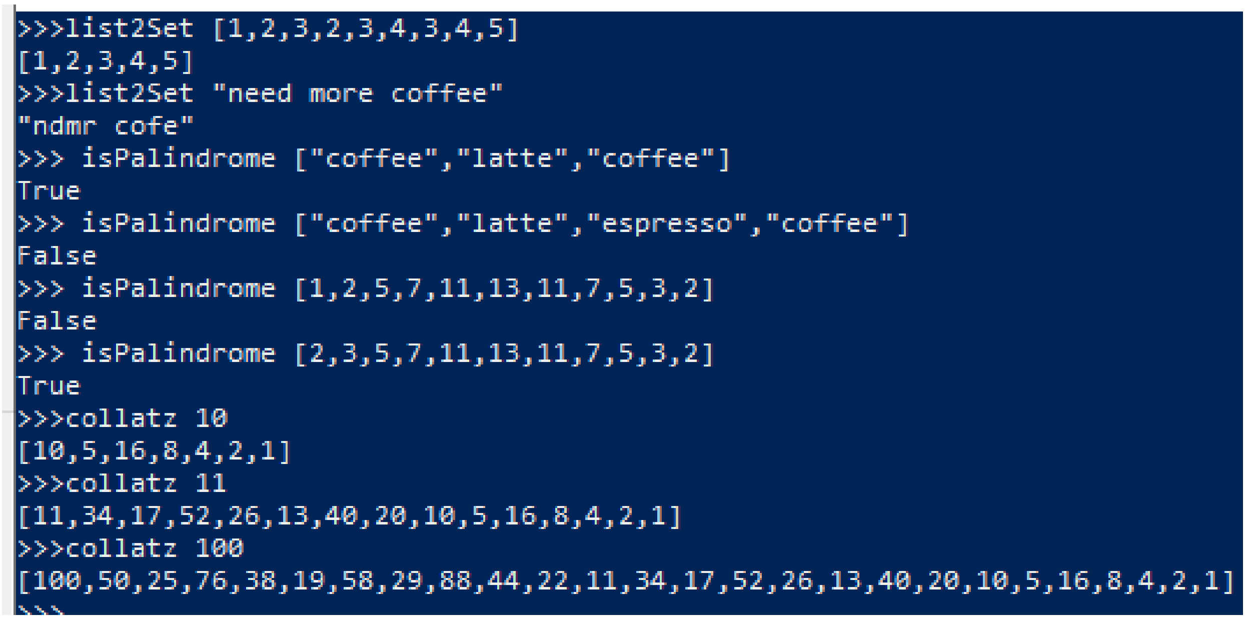
```
-----------------------------------------------------------------------------------
-- reverseWords
reverseWords :: String -> String

reverseWords characterString = unwords (reverse (words characterString))

-----------------------------------------------------------------------------------
-- averageWordLength
averageWordLength :: Fractional a => [Char] -> a

averageWordLength characterString = fromIntegral numberCharacters / fromIntegral numberWords
   where numberCharacters = length(filter (/=' ') characterString)
         numberWords = length(words characterString)

-----------------------------------------------------------------------------------
```



```
>>>reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
>>>reverseWords "want me some coffee"
"coffee some me want"
>>>reverseWords "fun was game hockey the"
"the hockey game was fun"
>>>reverseWords "let's have pancakes for dinner"
"dinner for pancakes have let's"
>>>averageWordLength "appa and baby yoda are the best"
3.5714285714285716
>>>averageWordLength "want me some coffee"
4.0
>>>averageWordLength "fun was game hockey the"
3.8
>>>averageWordLength "let's have pancakes for dinner"
5.2
```

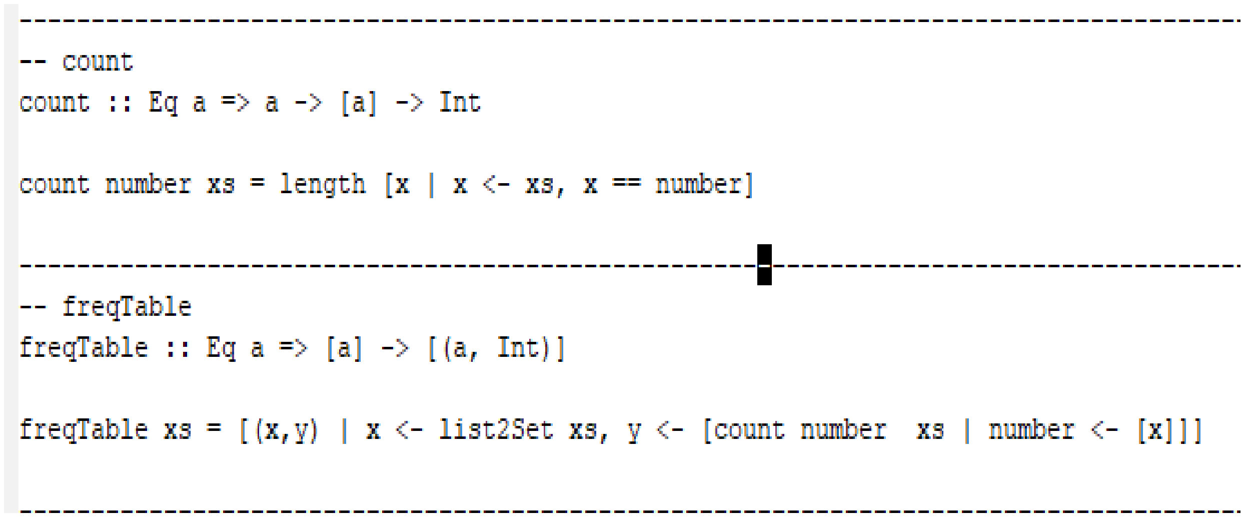
Task 4 – Recursive List Processors

```
---------------------------------------------------------------------------
-- list2Set
list2Set :: Eq a => [a] -> [a]

list2Set [] = []
list2Set (x:xs) = if (elem x xs) then (list2Set xs) else (x: list2Set xs)
---------------------------------------------------------------------------
-- isPalindrome
isPalindrome :: Eq a => [a] -> Bool

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

---------------------------------------------------------------------------
-- collatz
collatz :: Int -> [Int]

collatz 1 = [1]
collatz num = if ( odd num ) then (num : collatz ((3 * num) + 1))
                 else (num: collatz (div num 2))

---------------------------------------------------------------------------
```

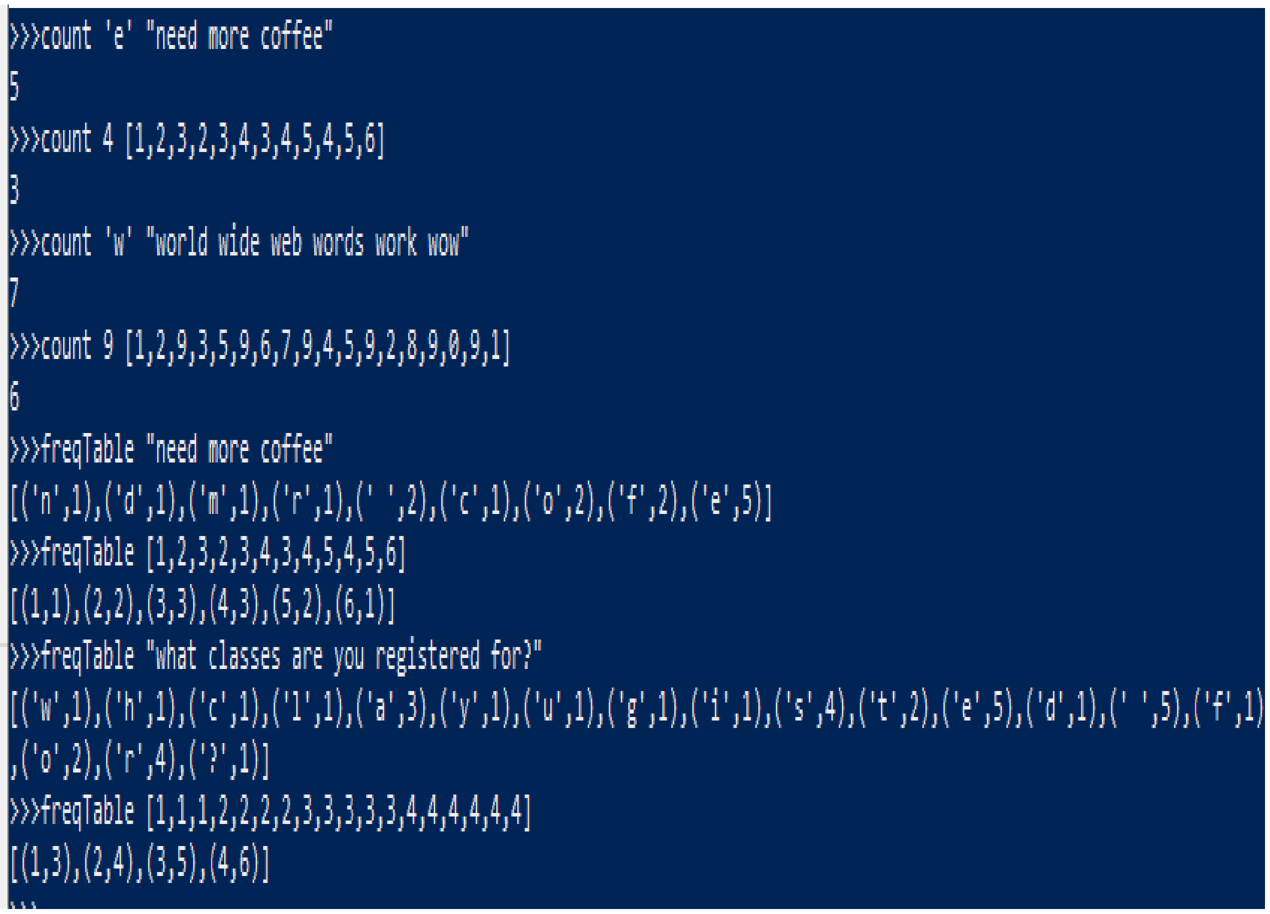


```
>>>list2Set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>>list2Set "need more coffee"
"ndmr cofe"
>>> isPalindrome ["coffee","latte","coffee"]
True
>>> isPalindrome ["coffee","latte","espresso","coffee"]
False
>>> isPalindrome [1,2,5,7,11,13,11,7,5,3,2]
False
>>> isPalindrome [2,3,5,7,11,13,11,7,5,3,2]
True
>>>collatz 10
[10,5,16,8,4,2,1]
>>>collatz 11
[11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
>>>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]
>>>
```

Task 5 – List Comprehensions



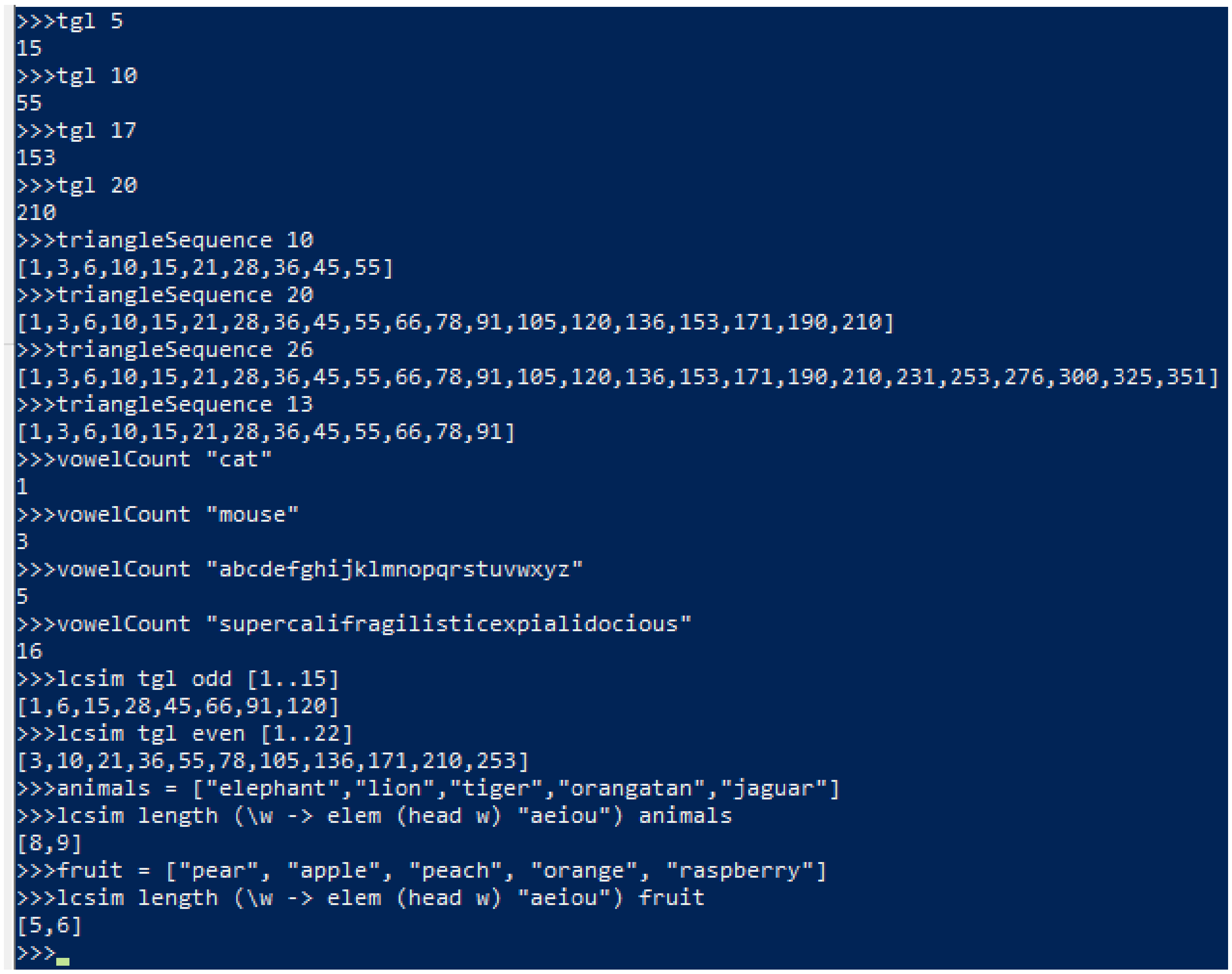
```
-----------------------------------------------------------------------------------
-- count
count :: Eq a => a -> [a] -> Int

count number xs = length [x | x <- xs, x == number]

-----------------------------------------------------------------------------------
-- freqTable
freqTable :: Eq a => [a] -> [(a, Int)]

freqTable xs = [(x,y) | x <- list2Set xs, y <- [count number  xs | number <- [x]]]

-----------------------------------------------------------------------------------
```



```
>>>count 'e' "need more coffee"
5
>>>count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
3
>>>count 'w' "world wide web words work wow"
7
>>>count 9 [1,2,9,3,5,9,6,7,9,4,5,9,2,8,9,0,9,1]
6
>>>freqTable "need more coffee"
[('n',1),('d',1),('m',1),('r',1),(' ',2),('c',1),('o',2),('f',2),('e',5)]
>>>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)]
>>>freqTable "what classes are you registered for?"
[('w',1),('h',1),('c',1),('l',1),('a',3),('y',1),('u',1),('g',1),('i',1),('s',4),('t',2),('e',5),('d',1),(' ',5),('f',1)
,('o',2),('r',4),('?',1)]
>>>freqTable [1,1,1,2,2,2,2,3,3,3,3,3,4,4,4,4,4,4]
[(1,3),(2,4),(3,5),(4,6)]
```

Task 6 – Higher Order Functions

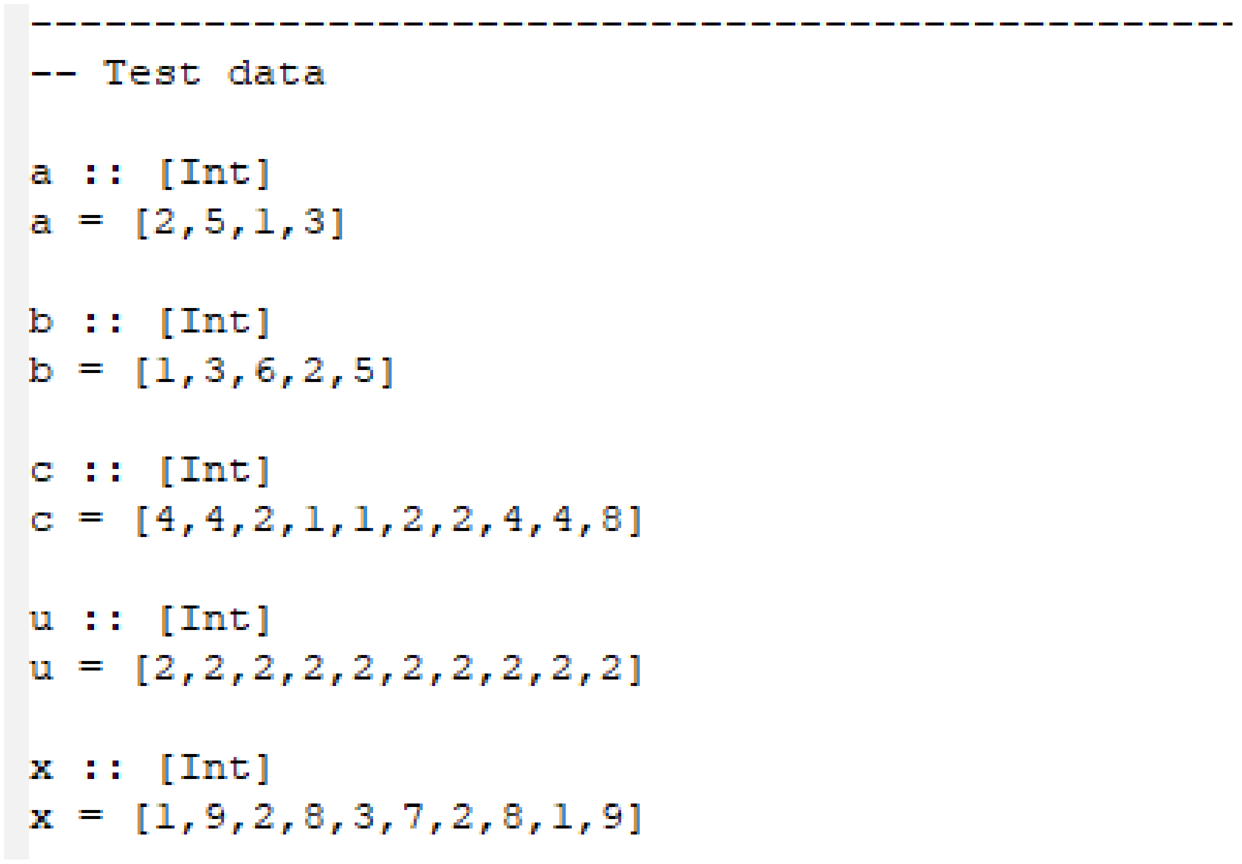
```
-------------------------------------------------------------------------------------------
-- tgl
tgl :: (Num b, Enum b) => b -> b

tgl number = foldl (+) 0 [1..number]

-------------------------------------------------------------------------------------------
-- trianlgeSequence
triangleSequence :: (Num b, Enum b) => b -> [b]

triangleSequence number = map (tgl) [1..number]

-------------------------------------------------------------------------------------------
-- vowelCount
vowelCount :: [Char] -> Int

vowelCount string = length (filter (\x -> (x `elem` ['a','e','i','o','u'])) string)

-------------------------------------------------------------------------------------------
-- lcsim
lcsim :: (a -> b) -> (a -> Bool) -> [a] -> [b]

lcsim function predicate list = map (function) (filter (predicate) list)
```



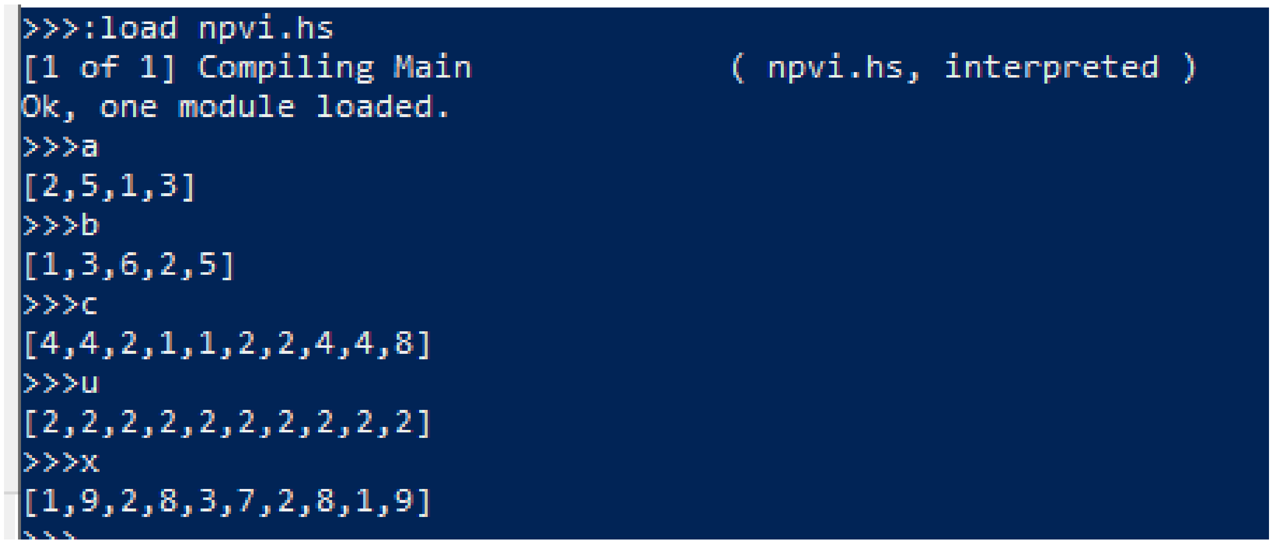
```
>>>tgl 5
15
>>>tgl 10
55
>>>tgl 17
153
>>>tgl 20
210
>>>triangleSequence 10
[1,3,6,10,15,21,28,36,45,55]
>>>triangleSequence 20
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210]
>>>triangleSequence 26
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210,231,253,276,300,325,351]
>>>triangleSequence 13
[1,3,6,10,15,21,28,36,45,55,66,78,91]
>>>vowelCount "cat"
1
>>>vowelCount "mouse"
3
>>>vowelCount "abcdefghijklmnopqrstuvwxyz"
5
>>>vowelCount "supercalifragilisticexpialidocious"
16
>>>lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
>>>lcsim tgl even [1..22]
[3,10,21,36,55,78,105,136,171,210,253]
>>>animals = ["elephant","lion","tiger","orangatan","jaguar"]
>>>lcsim length (\w -> elem (head w) "aeiou") animals
[8,9]
>>>fruit = ["pear", "apple", "peach", "orange", "raspberry"]
>>>lcsim length (\w -> elem (head w) "aeiou") fruit
[5,6]
>>>
```

Task 7 – An Interesting Statistic: nPVI

Task 7a – Test Data



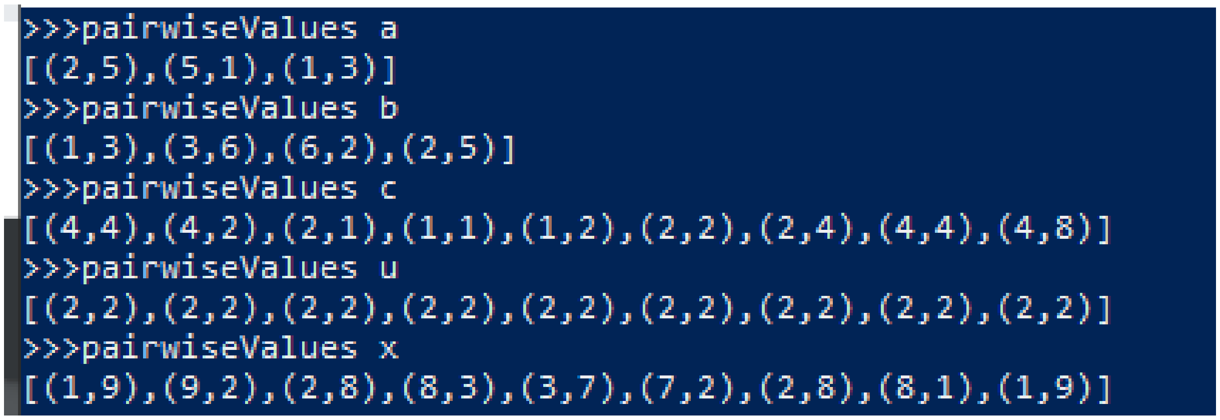
```
-----------------------------------------------------
-- Test data

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]
```



```
>>>:load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>>a
[2,5,1,3]
>>>b
[1,3,6,2,5]
>>>c
[4,4,2,1,1,2,2,4,4,8]
>>>u
[2,2,2,2,2,2,2,2,2,2]
>>>x
[1,9,2,8,3,7,2,8,1,9]
>>>
```

Task 7b – The pairwiseValues function

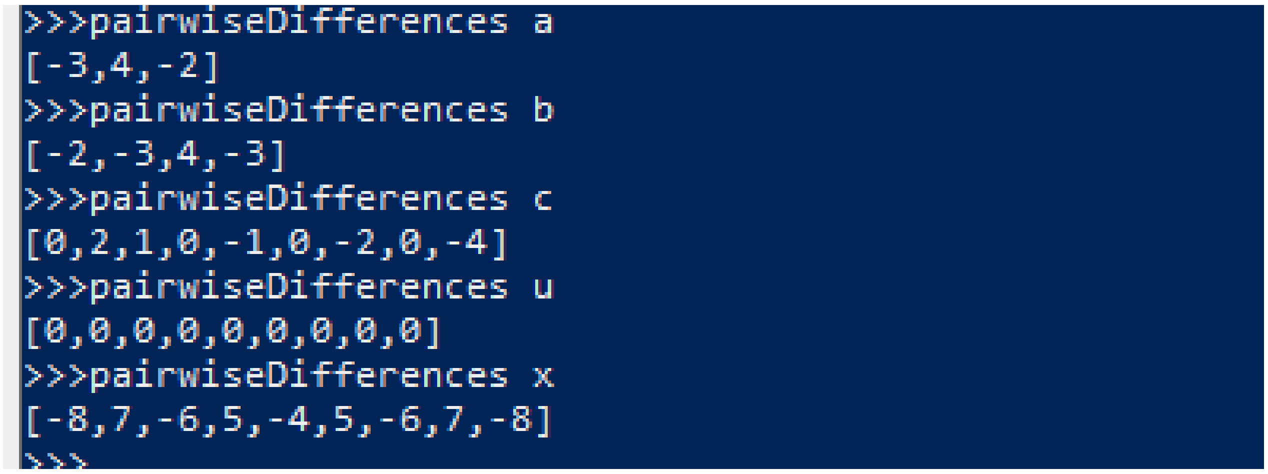
```
--------------------------------------------------------------------------------
-- pairwiseValues
pairwiseValues :: [Int] -> [(Int,Int)]

pairwiseValues (x:xs) = zip (x:xs) xs


--------------------------------------------------------------------------------
```



```
>>>pairwiseValues a
[(2,5),(5,1),(1,3)]
>>>pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
>>>pairwiseValues c
[(4,4),(4,2),(2,1),(1,1),(1,2),(2,2),(2,4),(4,4),(4,8)]
>>>pairwiseValues u
[(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2)]
>>>pairwiseValues x
[(1,9),(9,2),(2,8),(8,3),(3,7),(7,2),(2,8),(8,1),(1,9)]
```

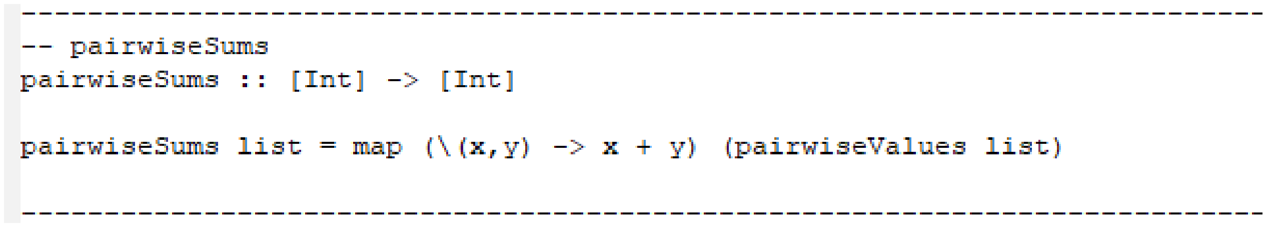
Task 7c – The pairwiseDifferences function

```
--------------------------------------------------------------------------------
-- pairwiseDifferences
pairwiseDifferences :: [Int] -> [Int]

pairwiseDifferences list = map (\(x,y) -> x - y) (pairwiseValues list)

--------------------------------------------------------------------------------
```

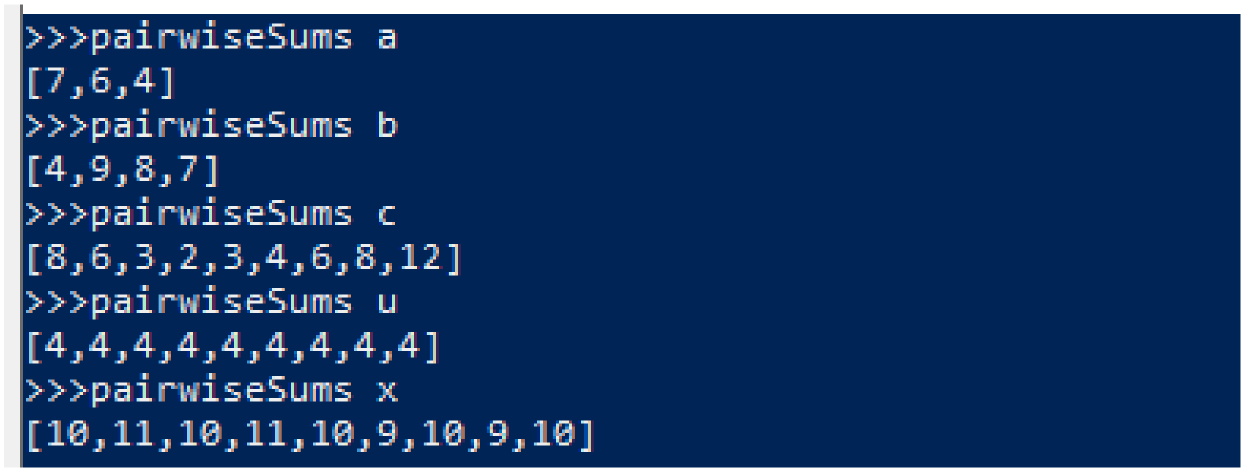


```
>>>pairwiseDifferences a
[-3,4,-2]
>>>pairwiseDifferences b
[-2,-3,4,-3]
>>>pairwiseDifferences c
[0,2,1,0,-1,0,-2,0,-4]
>>>pairwiseDifferences u
[0,0,0,0,0,0,0,0,0]
>>>pairwiseDifferences x
[-8,7,-6,5,-4,5,-6,7,-8]
>>>
```

Task 7d – The pairwiseSums function

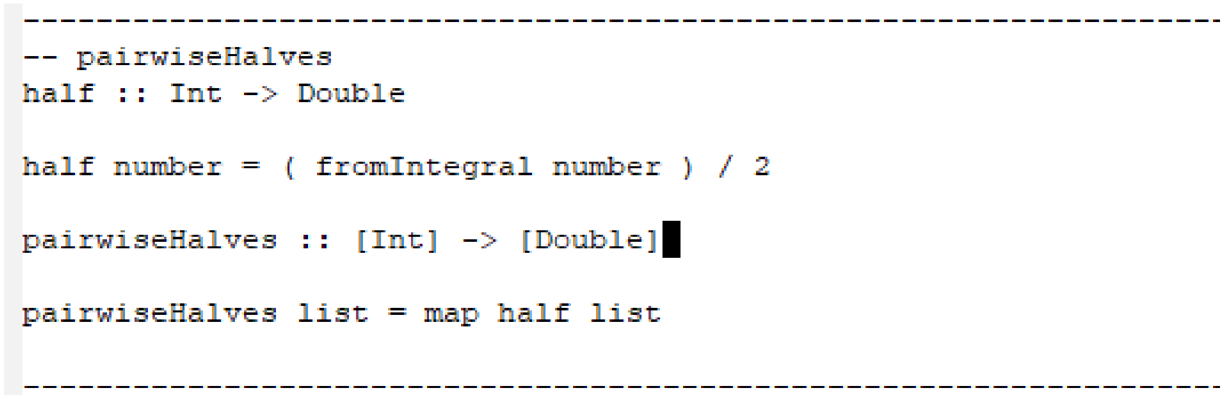


```
--------------------------------------------------------------------------------
-- pairwiseSums
pairwiseSums :: [Int] -> [Int]

pairwiseSums list = map (\(x,y) -> x + y) (pairwiseValues list)

--------------------------------------------------------------------------------
```

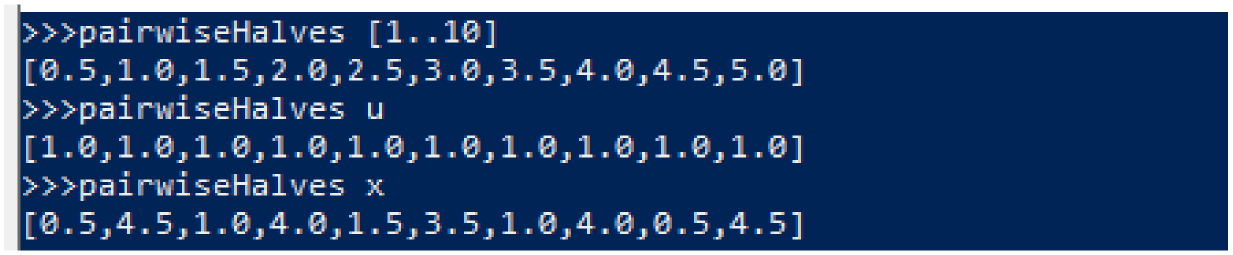


```
>>>pairwiseSums a
[7,6,4]
>>>pairwiseSums b
[4,9,8,7]
>>>pairwiseSums c
[8,6,3,2,3,4,6,8,12]
>>>pairwiseSums u
[4,4,4,4,4,4,4,4,4]
>>>pairwiseSums x
[10,11,10,11,10,9,10,9,10]
```

Task 7e – The pairwiseHalves function



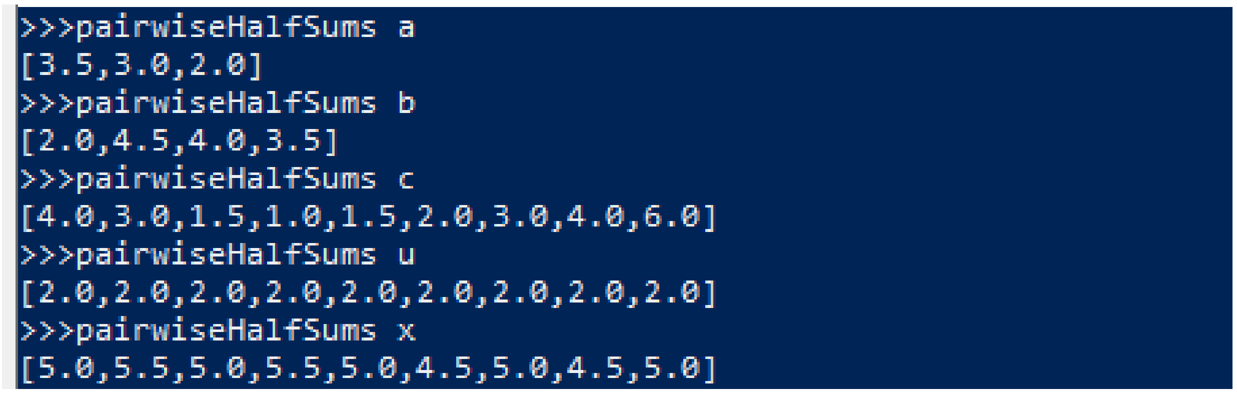
```
-----------------------------------------------------------------
-- pairwiseHalves
half :: Int -> Double

half number = ( fromIntegral number ) / 2

pairwiseHalves :: [Int] -> [Double]

pairwiseHalves list = map half list

-----------------------------------------------------------------
```



```
>>>pairwiseHalves [1..10]
[0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
>>>pairwiseHalves u
[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
>>>pairwiseHalves x
[0.5,4.5,1.0,4.0,1.5,3.5,1.0,4.0,0.5,4.5]
```

Task 7f – The pairwiseHalfSums function

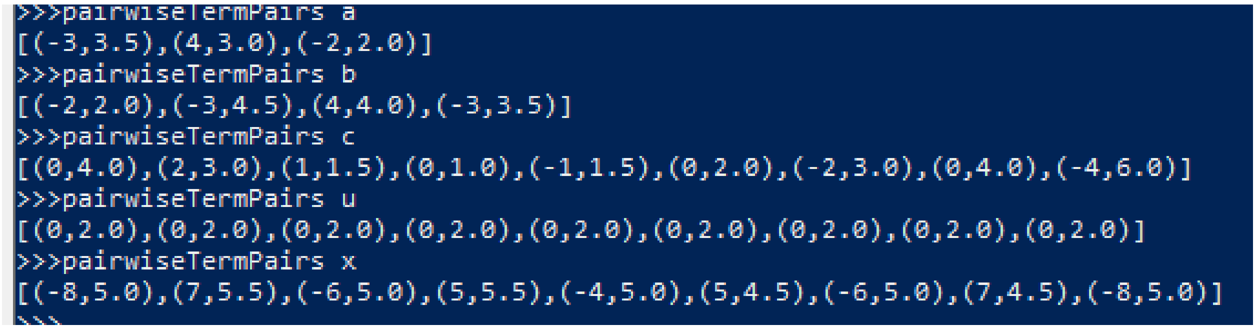
```
--------------------------------------------------------------------
-- pairwiseHalfSums

pairwiseHalfSums :: [Int] -> [Double]

pairwiseHalfSums list = pairwiseHalves (pairwiseSums list)

--------------------------------------------------------------------
```



```
>>>pairwiseHalfSums a
[3.5,3.0,2.0]
>>>pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
>>>pairwiseHalfSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,6.0]
>>>pairwiseHalfSums u
[2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0]
>>>pairwiseHalfSums x
[5.0,5.5,5.0,5.5,5.0,4.5,5.0,4.5,5.0]
```

Task 7g – The pairwiseTermPairs function

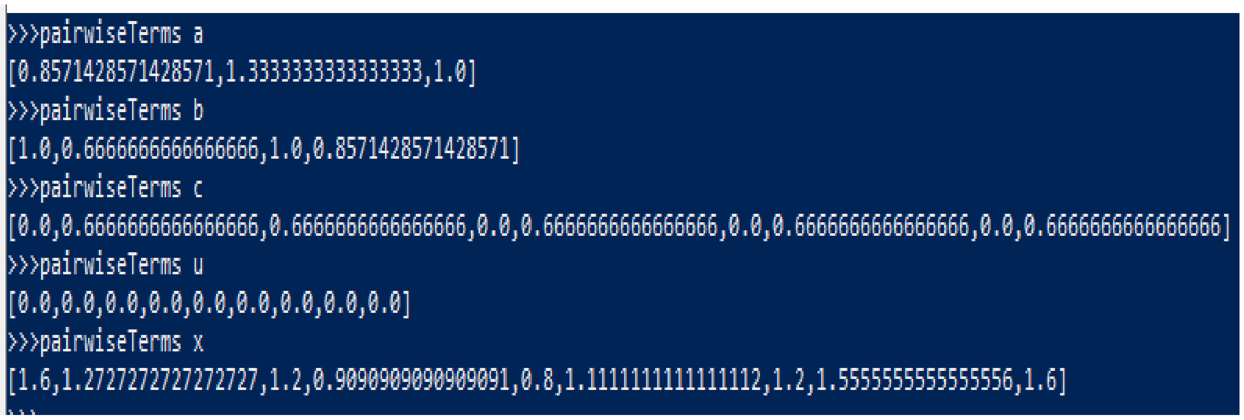
```
----------------------------------------------------------------------------------
-- pairwiseTermPairs

pairwiseTermPairs :: [Int] -> [(Int,Double)]

pairwiseTermPairs list = zip (pairwiseDifferences list) (pairwiseHalfSums list)

----------------------------------------------------------------------------------
```



```
>>>pairwiseTermPairs a
[(-3,3.5),(4,3.0),(-2,2.0)]
>>>pairwiseTermPairs b
[(-2,2.0),(-3,4.5),(4,4.0),(-3,3.5)]
>>>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)]
>>>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)]
>>>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)]
>>>
```

Task 7h – The pairwiseTerms function

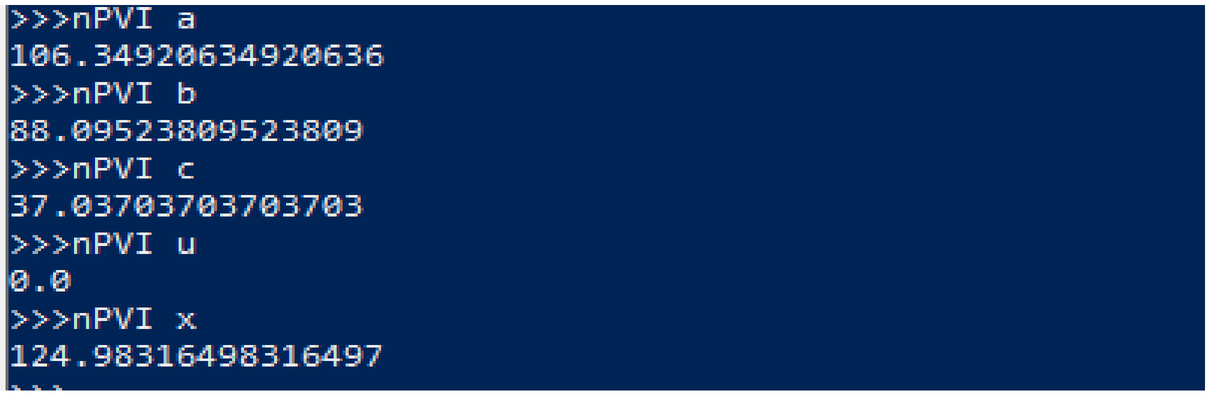
```
--------------------------------------------------------------------------
-- pairwiseTerms

term :: (Int,Double) -> Double

term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )

pairwiseTerms :: [Int] -> [Double]

pairwiseTerms list = map term (pairwiseTermPairs list)

--------------------------------------------------------------------------
```



```
>>>pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
>>>pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
>>>pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666]
>>>pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
>>>pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555555555555556,1.6]
```

Task 7i – The nPVI function

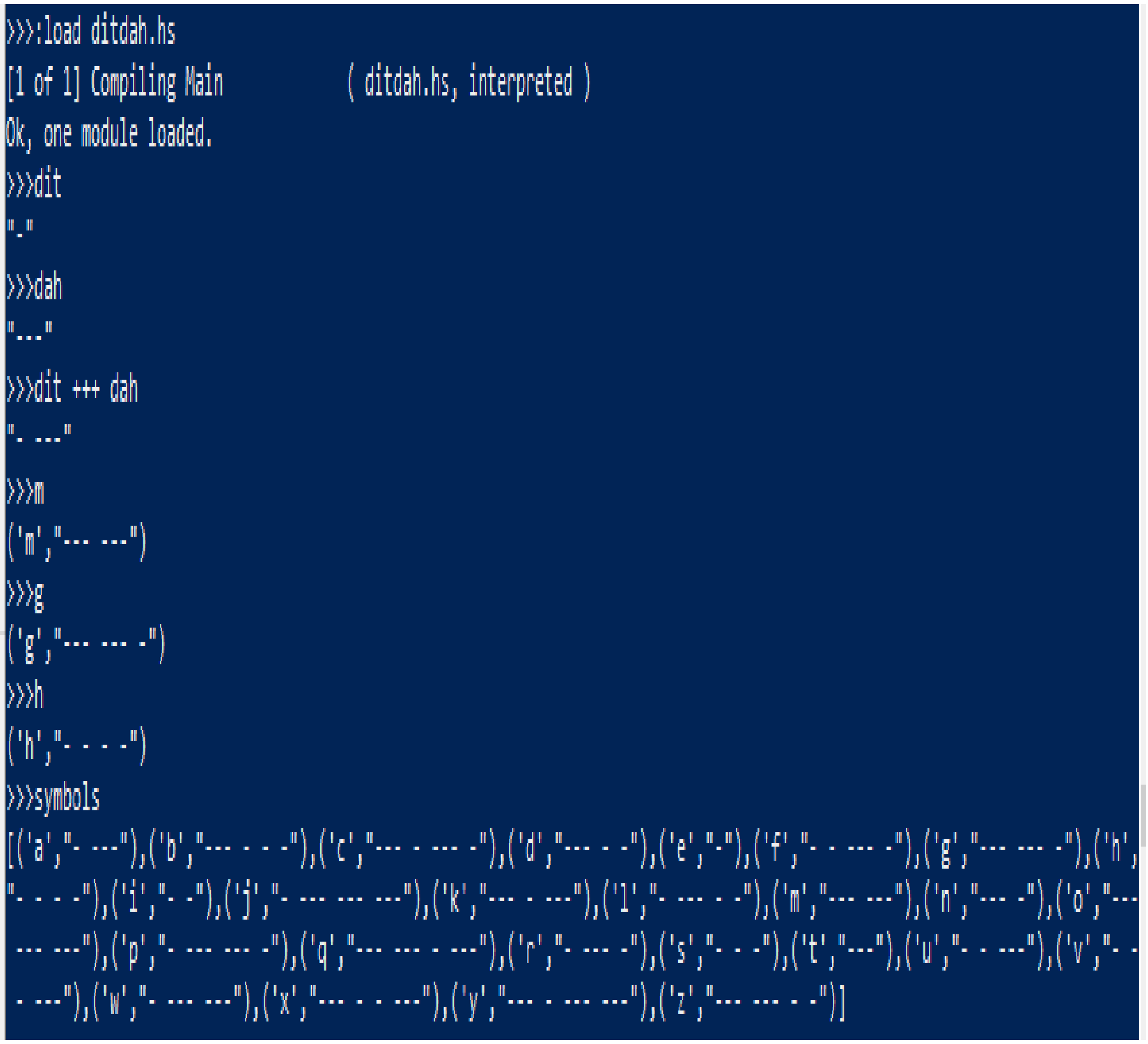
```
--------------------------------------------------------------------------
-- nPVI

nPVI :: [Int] -> Double

nPVI xs = normalizer xs * sum ( pairwiseTerms xs )
     where normalizer xs = 100 / fromIntegral ( ( length xs ) - 1 )
```



```
>>>nPVI a
106.34920634920636
>>>nPVI b
88.09523809523809
>>>nPVI c
37.03703703703703
>>>nPVI u
0.0
>>>nPVI x
124.98316498316497
```

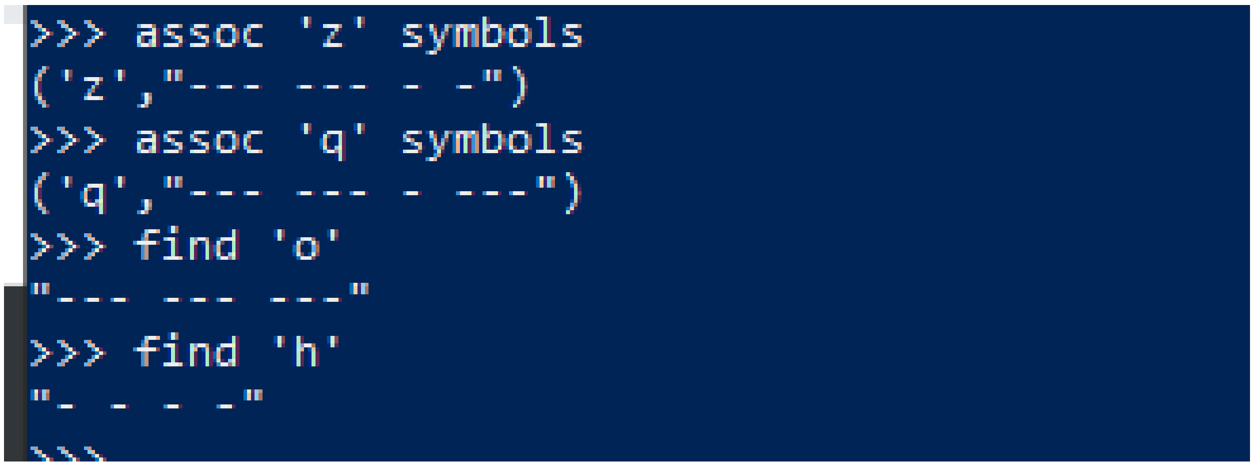
Task 8 – Historic Code: The Dit Dah Code

Subtask 8a



```
>>>:load ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
>>>dit
"-"
>>>dah
"---"
>>>dit +++ dah
"- ---"
>>>m
('m',"--- ---")
>>>g
('g',"--- --- -")
>>>h
('h',"- - - -")
>>>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',"--- --- - -")]
```

Subtask 8b



```
>>> assoc 'z' symbols
('z',"--- --- - -")
>>> assoc 'q' symbols
('q',"--- --- - ---")
>>> find 'o'
"--- --- ---"
>>> find 'h'
"- - - -"
>>>
```