Haskell Programming Assignment: Various Computations

Written by **David Hennigan**

Learning Abstract

This assignment played an important role in cultivating my understanding of Haskell. Throughout this assignment 8 tasks were completed each sequential task requiring slightly more knowledge of Haskell. Task 7 was my favorite task of this assignment, I got to use Haskell to build a mathematical function (nPVI) from the ground up.

Task 1 – Mindfully Mimicking the Demo

Demo

```
david@david-IdeaPad-1-15ALC7 ~ ghci
GHCi, version 8.8.4: 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?"
>>> :quit
Leaving GHCi.
david@david-IdeaPad-1-15ALC7 ~
```

Task 2 – Numeric Function Definitions

Function Definitions

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

--- ha.hs houses task 2 of the haskell assignment


--------------------------------------------------------------------------

--- squareArea :: takes one real number representing the side length

--- of a square and returns the area of the square with the given side

--- length


squareArea x = x * x


--------------------------------------------------------------------------

--- circleArea :: takes one real number representing the radius of a circle

--- and returns the corresponding circles area


circleArea r = pi * r ^ 2


--------------------------------------------------------------------------

--- blueAreaOfCube :: takes the length of one edge of the cube and computes

--- the area of the cube that is equivalent to the cube's area minus a

--- centered dots area ( on each side of the cube )

--- with a radius 1/4 the edges length.


blueAreaOfCube el = (6 * el ^ 2) - (6 * pi * ( el / 4 ) ^ 2)



--------------------------------------------------------------------------

--- paintedCube1 :: takes the order of a n by n by n cube where n is
```

--- equal to order and returns the number of cubes that are painted once

--- when the entire cube is painted

paintedCube1 order = if order < 3 then 0 else 6 * ((order - 2) ^ 2)

--- paintedCube2 :: takes the order of a n by n by n cube where n is equal

--- to order and returns the number of cubes that are painted twice when the

--- entire cube is painted

paintedCube2 order = if order < 3 then 0 else 12 * (order - 2)

Demo

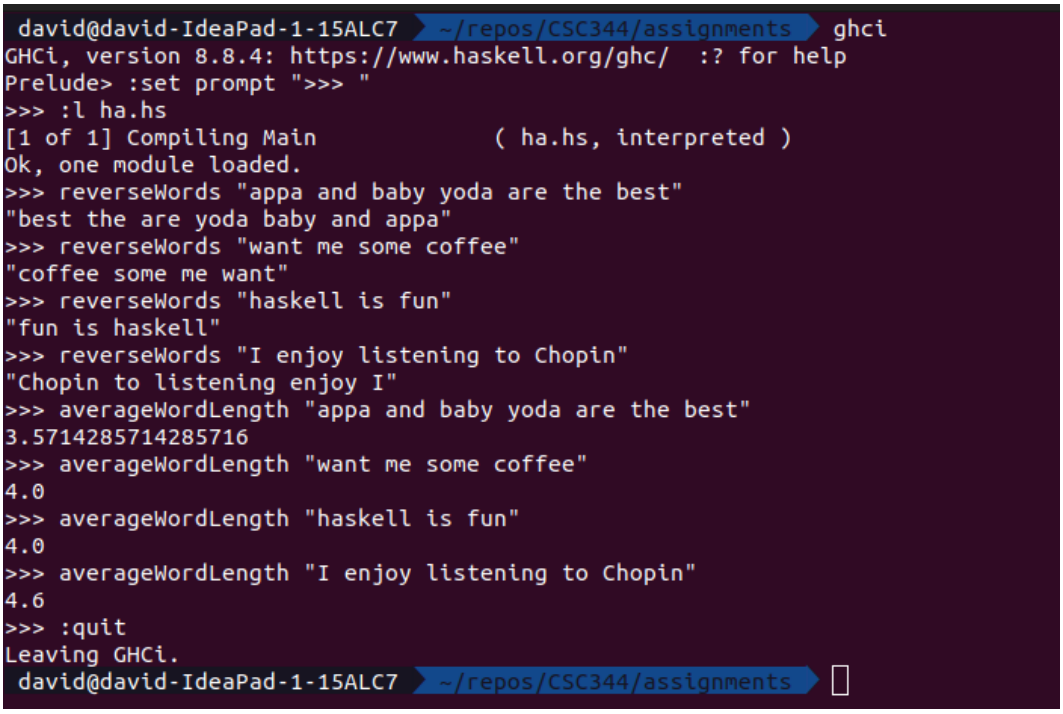
```
david@david-IdeaPad-1-15ALC7 ~/repos/CSC344/assignments ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :load ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> squareArea 10
100
>>> squareArea 12
144
>>> 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]
>>> :quit
Leaving GHCi.
david@david-IdeaPad-1-15ALC7 ~/repos/CSC344/assignments
```

Task 3 – Puzzlers

Function Definitions

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

--- reverseWords :: reverses the words of a string


reverseWords wordString = unwords ( reverse ( words wordString ) )


--------------------------------------------------------------------------

--- averageWordLength :: takes a string of words and returns the average

--- word length


averageWordLength wordString = fromIntegral ( sum letterCountList ) /  fromIntegral wordCount

   where wordList = words wordString

            wordCount = length wordList

            letterCountList = map length wordList
```

Demo



Task 4 – Recursive List Processors

Function Definitions

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

--- list2set :: takes a list and turns it into a set


list2set l = if (length l) == 0 then [] else (if elem (head l) set then set else (head l) : set)
    where set = list2set(tail l)


--------------------------------------------------------------------------

--- isPalindrome :: checks to see if a list is a palindrome


isPalindrome l = if (length l) < 2 then True else (if (head l) == (last l) then (isPalindrome (tail (init l)))
else False)


--------------------------------------------------------------------------

--- collatz :: takes a number as an input and generates a list containing

--- the corresponding collatz sequence


collatz :: Integer -> [Integer]

collatz num = if num == 1 then [1] else ( num : l )

    where nextNum = if ( rem num 2 ) == 0 then (div num 2) else (3 * num + 1)

          l = ( collatz nextNum )
```

Demo

```
 david@david-IdeaPad-1-15ALC7  ~/repos/CSC344/assignments  ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> 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]
>>> :quit
Leaving GHCi.
 david@david-IdeaPad-1-15ALC7  ~/repos/CSC344/assignments
```

Task 5 – List Comprehensions

Function Definitions

--- count :: takes an object and a list of objects then returns the amount

--- of times the object appears within the list.

count object objects = length [x | x <- objects, x == object]

--- freqTable :: takes a list of objects and returns a list of ordered

--- pairs each consisting of an element of the list together with the number

--- of times it occured

freqTable objects = [(x,y) | x <- (list2set objects), y <- [(count x objects)]]

Demo

```
 david@david-IdeaPad-1-15ALC7  ~/repos/CSC344/assignments  ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :load ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> count 'e' "need more coffee"
5
>>> count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
3
>>> count 'a' "lambdas are rad"
4
>>> count 17 [17,3,5,7,9,17,4]
2
>>> 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 "lambdas are rad"
[('l',1),('m',1),('b',1),('s',1),('e',1),(' ',2),('r',2),('a',4),('d',2)]
>>> freqTable [17,3,5,7,9,17,4]
[(3,1),(5,1),(7,1),(9,1),(17,2),(4,1)]
>>> :quit
Leaving GHCi.
 david@david-IdeaPad-1-15ALC7  ~/repos/CSC344/assignments
```

Task 6 – Higher Order Functions

Function Definitions

--- vowelCount :: takes a string of lower case letters and returns the number

--- of lowercase vowels that appear in the string

vowelCount str = length (filter (\x -> (x == 'a' || x == 'e' || x == 'i' ||

x == 'o' || x == 'u')) str)

--- lcsim :: takes a function for mapping, a predicate for filtering, and

--- a list of elements and returns the same value as:

--- [f x | x <- xs, p x]

lcsim mapFunc pred elems = map (mapFunc) (filter (pred) elems)

Demo

```
david@david-IdeaPad-1-15ALC7 ~/repos/CSC344/assignments ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :load ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> tgl 5
15
>>> tgl 10
55
>>> tgl 47
1128
>>> 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 47
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210,231,253,276,300,325,351,378,406,435,465,496,528,561,595,630,
666,703,741,780,820,861,903,946,990,1035,1081,1128]
>>> triangleSequence 5
[1,3,6,10,15]
>>> vowelCount "cat"
1
>>> vowelCount "mouse"
3
>>> vowelCount "rat"
1
>>> vowelCount "lambda"
2
>>> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
>>> animals = ["elephant","lion","tiger","orangutan","jaguar"]
>>> lcsim length (\w -> elem ( head w ) "aeiou") animals
[8,9]
>>> lcsim tgl even [1..15]
[3,10,21,36,55,78,105]
>>> lcsim length (\w -> elem ( head w ) "t") animals
[5]
>>> :quit
Leaving GHCi.
david@david-IdeaPad-1-15ALC7 ~/repos/CSC344/assignments
```

Task 7 – An Interesting Statistic: nPVI

Task 7a – Test data

Function Definitions

--- 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]

Demo

```
>>> 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

Function Definitions

--- pairwiseValues

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

pairwiseValues listOfInts = zip (init listOfInts) (tail listOfInts)

Demo

```
>>> 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

Function Definitions

--- pairwiseDifferences

pairwiseDifferences :: [Int] -> [Int]

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

Demo

```
>>> 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

Function Definitions

--- pairwiseSums

pairwiseSums :: [Int] -> [Int]

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

Demo

```
>>> 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

Function Definitions

--- half

half :: Int -> Double

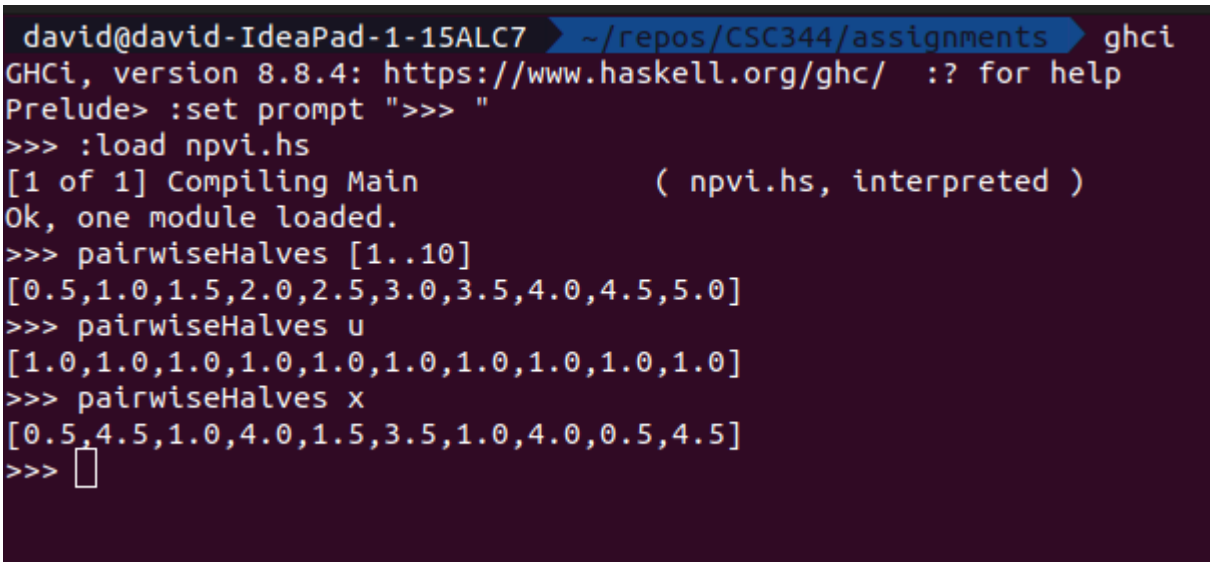
half number = (fromIntegral number) / 2

--- pairwiseHalves

pairwiseHalves :: [Int] -> [Double]

pairwiseHalves numbers = map (half) numbers

Demo



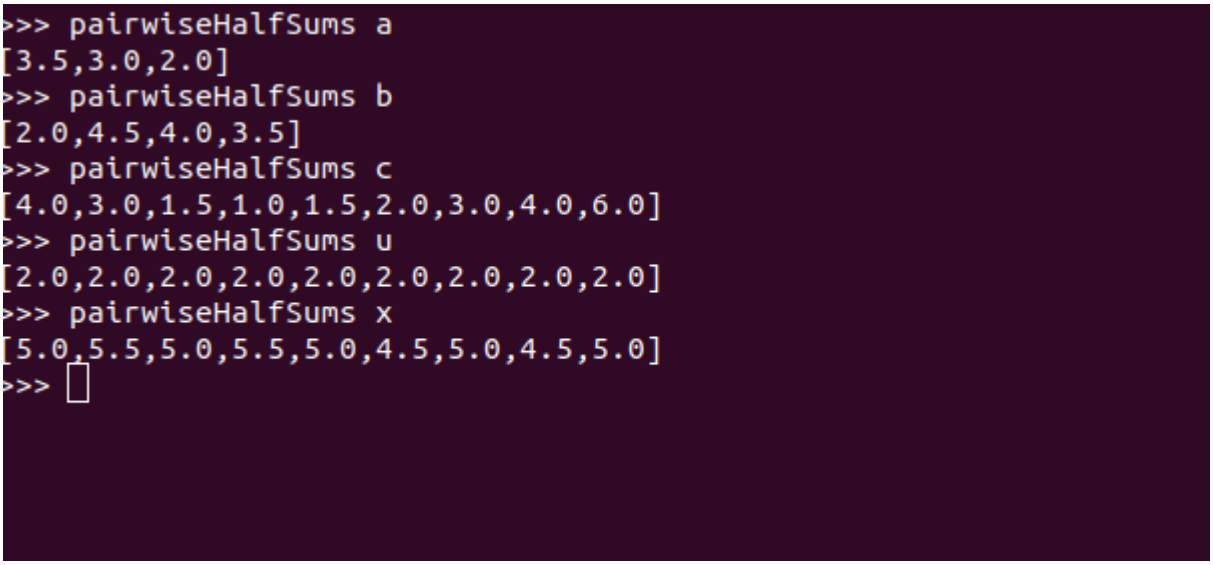
Task 7f – The pairwiseHalfSums function

Function Definition

--- pairwiseHalfSums

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

Demo



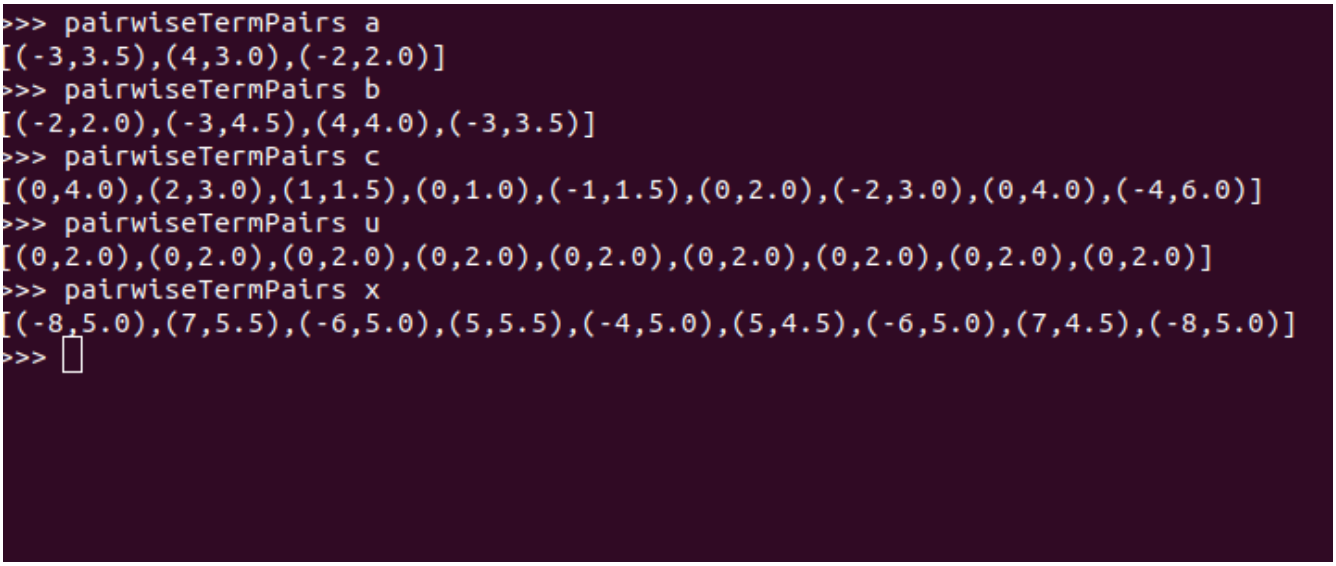
Task 7g – The pairwiseTermPairs function

Function Definition

--- pairwiseTermPairs

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

Demo



Task 7h – The pairwiseTerms functional

Function Definitions

--- term

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

--- pairwiseTerms

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

Demo

```
>>> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
>>> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
>>> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.66666666
66666666,0.0,0.6666666666666666]
>>> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
>>> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555
555555555556,1.6]
>>>
```

Task 7i – The nPVI function

Function Definition

--- nPVI

nPVI :: [Int] -> Double

nPVI xs = normalizer xs * sum (pairwiseTerms xs)

where normalizer xs = 100 / fromIntegral ((length xs) - 1)

Demo

```
>>> nPVI a
106.34920634920636
>>> nPVI b
38.09523809523809
>>> nPVI c
37.03703703703703
>>> nPVI u
0.0
>>> nPVI x
124.98316498316497
>>>
```

Task 8 – Historic Code: The Dit Dah Code

Subtask 8a

```
>>> dit
"-"
>>> dah
"---"
>>> (+++) "Hey" "there!"
"Hey there!"
>>> 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 'c' symbols
('c',"--- - --- -")
>>> assoc 'z' symbols
('z',"--- --- - -")
>>> find 'a'
"- ---"
>>> find 'b'
"--- - - -"
>>>
```

Subtask 8c

```
>>> addletter (find 'h') (find 'i')
"- - - -   - -"
>>> addword ( addletter (find 'h') (find 'i') ) ( addletter (find 'j') (find 'o') )
"- - - -   - -       - --- --- ---   --- --- ---"
>>> droplast3 (addword ( addletter (find 'h') (find 'i') ) ( addletter (find 'j') (find 'o') ) )
"- - - -   - -       - --- --- ---   --- --- "
>>> droplast7 (addword ( addletter (find 'h') (find 'i') ) ( addletter (find 'j') (find 'o') ) )
"- - - -   - -       - --- --- ---   --- "
>>>
```

Subtask 8d

```
>>> encodeletter 'm'
"--- ---"
>>> encodeletter 't'
"---"
>>> encodeletter 'u'
"- - ---"
>>> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
>>> encodeword "yo"
"--- - --- ---   --- --- ---"
>>> encodeword "awesome"
"- ---   - --- ---   -   - - -   --- --- ---   --- ---   -"
>>> encodemessage "need more coffee"
"--- -   -   -   --- - -       --- ---   --- --- ---   - --- -   -       --- - --- -   --- --- ---   - - -
-- -   - - --- -   -   -"
>>> encodemessage "high order functions are the best"
"- - - -   - -   --- --- -   - - - -       --- --- ---   - --- -   --- - -   -   - --- -       - - --- -
 - - ---   --- -   --- - --- -   ---   - -   --- --- ---   --- -   - - -       - ---   - --- -   -       -
--   - - - -   -       --- - - -   -   - - -   ---"
>>> encodemessage "this semester has been a blast"
"---   - - - -   - -   - - -       - - -   -   --- ---   -   - - -   ---   -   - --- -       - - - -   - -
--   - - -       --- - - -   -   -   --- -       - ---       --- - - -   - --- - -   - ---   - - -   ---"
>>>
```