Haskell Programming Assignment: Various Computations

Learning Abstract

For my first Haskell Programming Assignment, there are eight tasks in total which are based on basic functions, recursive list processing, higher order functions, and list comprehensions in Haskell.

Task 1 - Mindfully Mimicking the Demo

```
Last login: Wed Nov 16 11:02:14 on ttys000
lamin@Las-MacBook-Air ~ % ghci
GHCi, version 9.2.4: 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?"
>>> :quit
Leaving GHCi.
lamin@Las-MacBook-Air ~ %
```

Task 2 - Numeric Function Definitions

Haskell Code for Numeric Function Definitions

numeric_functions.hs

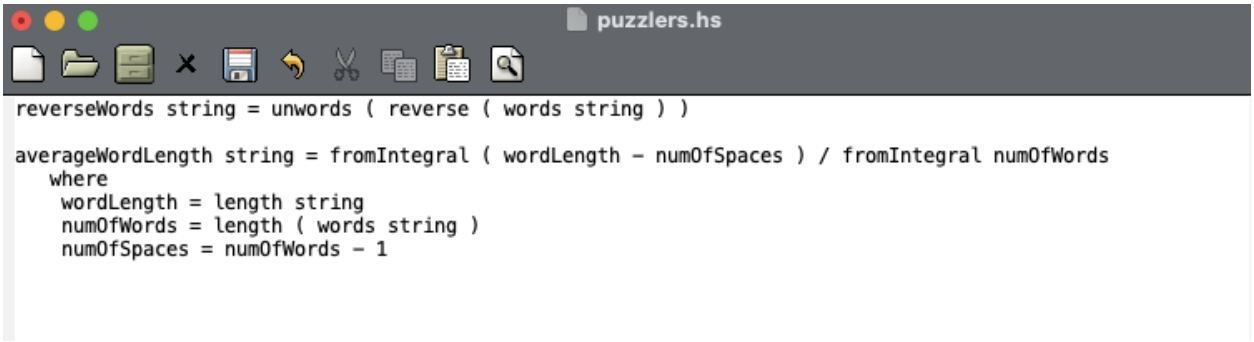
```
squareArea number = number * number

circleArea radius =  pi * radius^2

blueAreaOfCube edge = 6 * ( ( squareArea edge ) - ( circleArea ( edge / 4 ) ) )

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

paintedCube2 order = if ( order > 2 ) then ( 6 * ( 2 * ( order - 2 ) ) ) else 0
```

Demo for Numeric Function Definitions

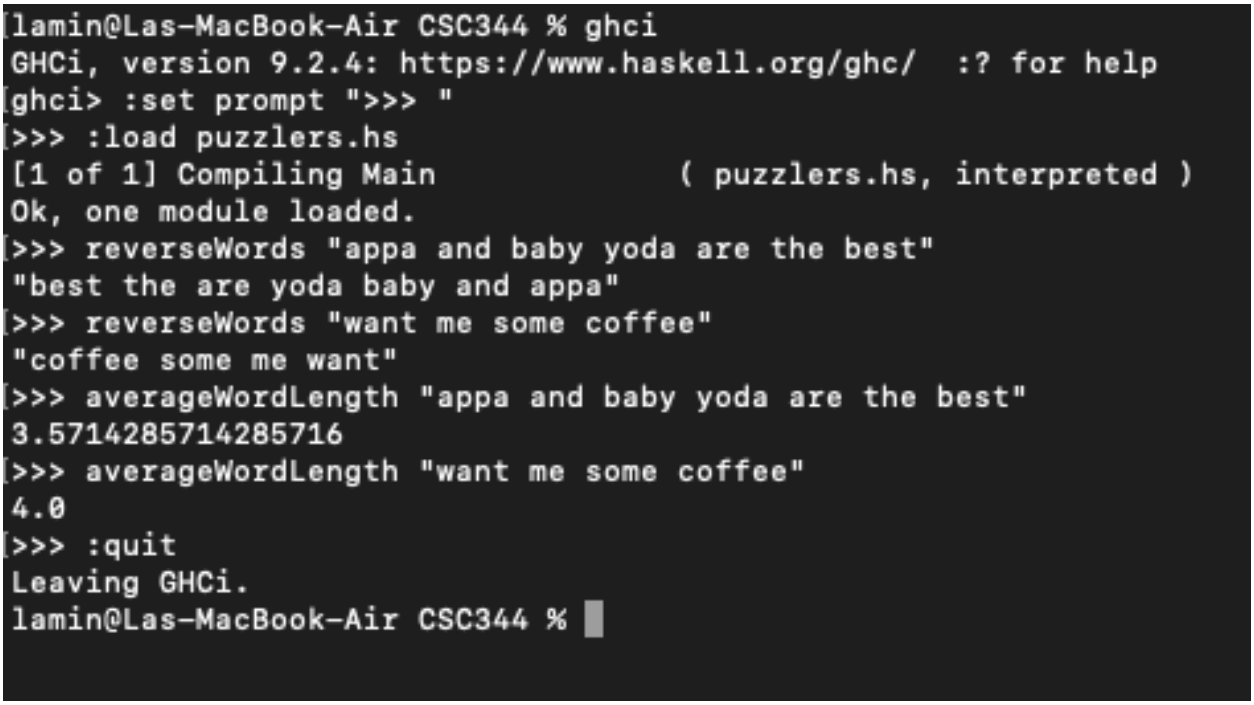
```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load numeric_functions.hs
[1 of 1] Compiling Main             ( numeric_functions.hs, interpreted )
Ok, one module loaded.
>>> squareArea 10
100
>>> squareArea 12
144
>>> 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.
lamin@Las-MacBook-Air CSC344 %
```

Task 3 - Puzzlers

Haskell Code for Puzzlers



Demo for Puzzlers



Task 4 - Recursive List Processors

Haskell Code for Recursive List Processors

Rlps.hs

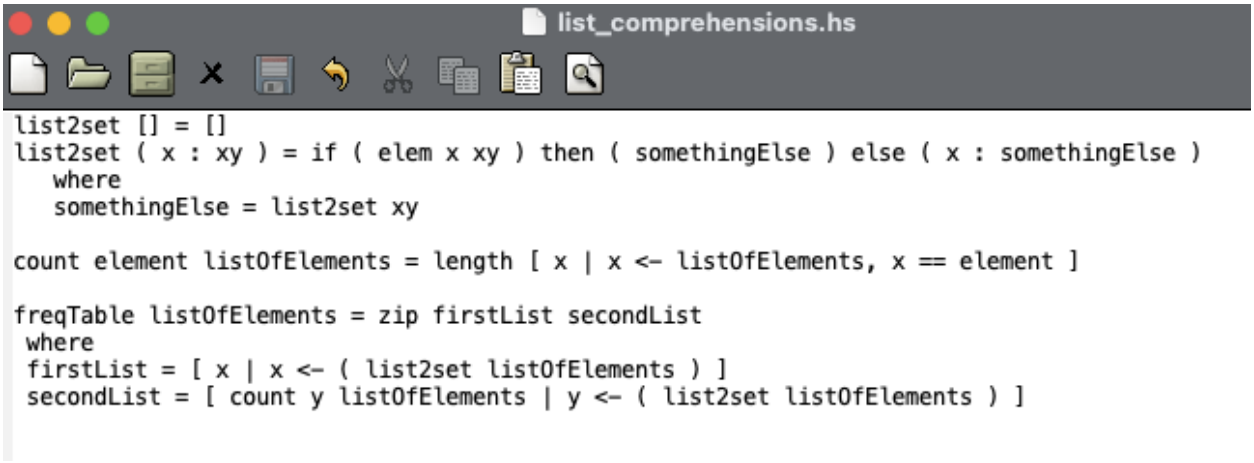
```
list2set [] = []
list2set ( x : xy ) = if ( elem x xy ) then ( somethingElse ) else ( x : somethingElse )
   where
   somethingElse = list2set xy


isPalindrome [] = True
isPalindrome ( x : xy ) = if ( length ( x : xy ) ) == 1 then True else ( if ( x == last xy ) then ( somethingElse ) else False )
 where
 somethingElse = isPalindrome ( head xy : drop 1 ( init xy ) )

collatz::Int->[Int]
collatz 1 = [1]
collatz number = if ( even number ) then ( number : collatz ( number `div` 2 ) ) else ( number : collatz ( ( 3*number ) + 1 ) )
```

Demo for Recursive List Processors

```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Rlps.hs
[1 of 1] Compiling Main             ( Rlps.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.
lamin@Las-MacBook-Air CSC344 %
```

Task 5 - List Comprehensions

Haskell Code for List Comprehensions

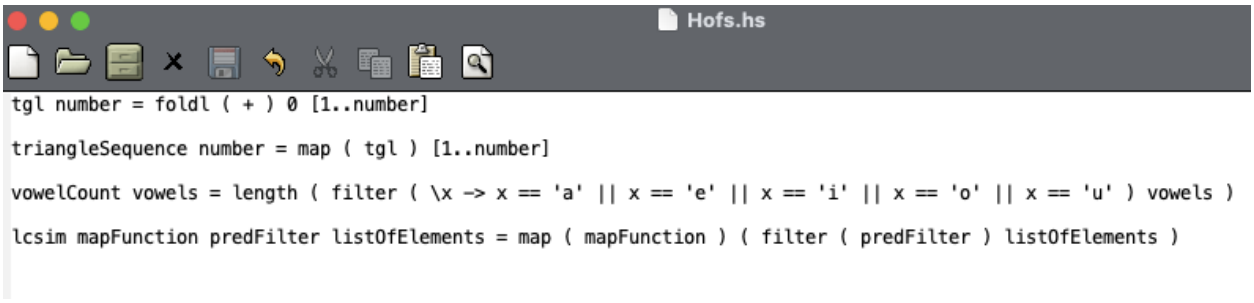


Demo for List Comprehensions

```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load list_comprehensions.hs
[1 of 1] Compiling Main             ( list_comprehensions.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
>>> 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)]
>>> :quit
Leaving GHCi.
lamin@Las-MacBook-Air CSC344 %
```

Task 6 - Higher Order Functions

Haskell Code for Higher Order Functions



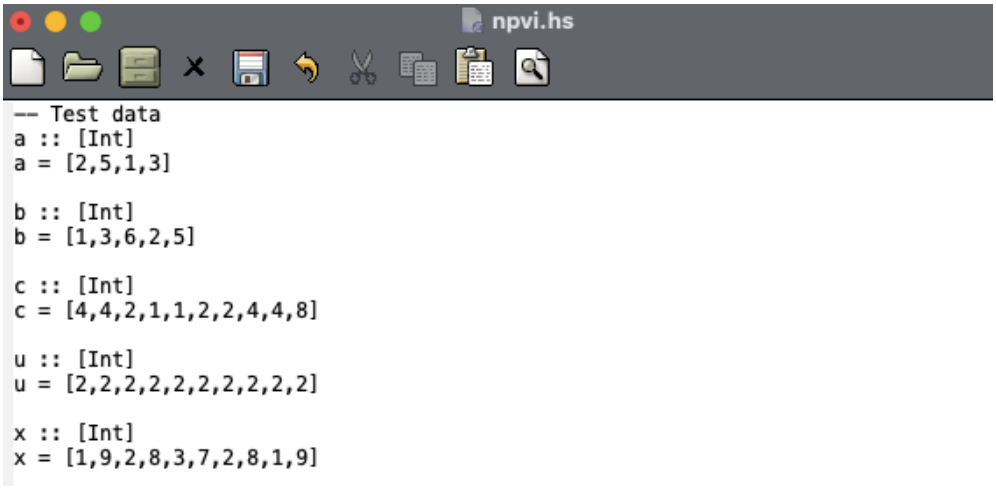
Demo for Higher Order Functions

```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Hofs.hs
[1 of 1] Compiling Main             ( Hofs.hs, interpreted )
Ok, one module loaded.
>>> tgl 5
15
>>> tgl 10
55
>>> 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]
>>> vowelCount "cat"
1
>>> vowelCount "mouse"
3
>>> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
>>> animals = ["elephant","lion","tiger","orangatan","jaguar"]
>>> lcsim length (\w -> elem ( head w ) "aeiou") animals
[8,9]
>>> :quit
Leaving GHCi.
lamin@Las-MacBook-Air CSC344 %
```

Task 7 - An Interesting Statistic: nPVI

Task 7a - Test Data

Haskell Code for Test Data



Demo for Test Data

```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :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]
>>> :quit
Leaving GHCi.
lamin@Las-MacBook-Air CSC344 %
```

Task 7b - The `pairwiseValues` function

Haskell Code for The `pairwiseValues` function

```
pairwiseValues :: [Int] -> [(Int,Int)]
pairwiseValues listOfNumbers = zip listOfNumbers ( tail listOfNumbers )
```

Demo for The `pairwiseValues` function

```
lamin@Las-MacBook-Air CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>> 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)]
>>> :quit
Leaving GHCi.
lamin@Las-MacBook-Air CSC344 %
```

Task 7c - The `pairwiseDifferences` function

Haskell Code for The `pairwiseDifferences` function

```
pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences listOfNumbers = map ( \ ( a,b ) -> a - b ) ( pairwiseValues listOfNumbers )
```

Demo for The `pairwiseDifferences` function

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>> 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]
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7d - The `pairwiseSums` function

Haskell Code for The `pairwiseSums` function

```
pairwiseSums :: [Int] -> [Int]
pairwiseSums listOfNumbers = map ( \ ( a,b ) -> a + b ) ( pairwiseValues listOfNumbers )
```

Demo for The `pairwiseSums` function

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>> 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]
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7e - The `pairwiseHalves` function

Haskell Code for The `pairwiseHalves` function

```
half :: Int -> Double
half number  = ( fromIntegral number ) / 2

pairwiseHalves :: [Int] -> [Double]
pairwiseHalves listOfNumbers = map half listOfNumbers
```

Demo for The `pairwiseHalves` function

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>> 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]
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7f - The `pairwiseHalfSums` function

Haskell Code for The `pairwiseHalfSums` function

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

Demo for The `pairwiseHalfSums` function

```
[lamin@res-dhcp-129-3-137-12 CSC344 % ghci
 GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
[ghci> :set prompt ">>> "
[>>> :load npvi.hs
 [1 of 1] Compiling Main             ( npvi.hs, interpreted )
 Ok, one module loaded.
[>>> 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]
[>>> :quit
 Leaving GHCi.
 lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7g - The `pairwiseTermPairs` function

Haskell Code for The `pairwiseTermPairs` function

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

Demo for The `pairwiseTermPairs` function

```
[lamin@res-dhcp-129-3-137-12 CSC344 % ghci
 GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
[ghci> :set prompt ">>> "
[>>> :load npvi.hs
 [1 of 1] Compiling Main             ( npvi.hs, interpreted )
 Ok, one module loaded.
[>>> 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)]
[>>> :quit
 Leaving GHCi.
 lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7h - The `pairwiseTerms` function

Haskell Code for The `pairwiseTerms` function

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

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

Demo for The `pairwiseTerms` function

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
>>> 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]
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 7i - The `nPVI` function

Haskell Code for The `nPVI` function

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

Demo for The `nPVI` function

```
[lamin@res-dhcp-129-3-137-12 CSC344 % ghci
 GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
[ghci> :set prompt ">>> "
[>>> :load npvi.hs
 [1 of 1] Compiling Main             ( npvi.hs, interpreted )
 Ok, one module loaded.
[>>> nPVI a
 106.34920634920636
[>>> nPVI b
 88.09523809523809
[>>> nPVI c
 37.03703703703703
[>>> nPVI u
 0.0
[>>> nPVI x
 124.98316498316497
[>>> :quit
 Leaving GHCi.
 lamin@res-dhcp-129-3-137-12 CSC344 %
```

Task 8 - Historic Code : The Dit Dah Code

Subtask 8a

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :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',"--- --- - -")]
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Subtask 8b

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
>>> assoc 'c' symbols
('c',"--- - --- -")
>>> assoc 'h' symbols
('h',"- - - -")
>>> find 'z'
"--- --- - -"
>>> find 'o'
"--- --- ---"
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Subtask 8c

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
>>> addletter "l" "s"
"l   s"
>>> addword "big" "cat"
"big       cat"
>>> droplast3 "Mathematics"
"Mathemat"
>>> droplast7 "New York City"
"New Yo"
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```

Subtask 8d

```
lamin@res-dhcp-129-3-137-12 CSC344 % ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'a'
"_ ___"
>>> encodeletter 'p'
"_ ___ ___ _"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "plane"
"_ ___ ___ _   _ ___ _ _   _ ___   ___ _   _"
>>> encodeword "home"
"_ _ _ _   ___ ___ ___   ___ ___   _"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _       ___ _ ___ _
   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
>>> encodemessage "i like computers"
"_ _       _ ___ _ _   _ _   ___ _ ___   _       ___ _ ___ _   ___ ___ ___   ___ ___
   _ ___ ___ _   _ _ ___   ___   _   _ ___ _   _ _ _"
>>> encodemessage "i saw a small bird"
"_ _       _ _ _   _ ___   _ ___ ___       _ ___       _ _ _   ___ ___   _ ___   _ _
__ _ _   _ ___ _ _       ___ _ _ _   _ _   _ ___ _   ___ _ _"
>>> :quit
Leaving GHCi.
lamin@res-dhcp-129-3-137-12 CSC344 %
```