Haskell Programming Assignment Solution

Task 1 - Mindfully Mimicking the Demo

```
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"]
>>> tail ["need","more","coffee"]
["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
>>> filter (\x -> x /= ' ') "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
```

Task 2 - Numeric Function Definitions

```
------ Task # 2 -------
squareArea n = n * n

circleArea n = n * n * pi

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

paintedCube1 order = 6 * ( ( order - 2 ) ^ 2 )

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

```
>>> squareArea 5
25
>>> circleArea 10
314.1592653589793
>>> blueAreaOfCube 1
4.821902754903828
>>> blueAreaOfCube 10
482.19027549038276
>>> blueAreaOfCube 210
482.19027549038276
```

Task 3 - Puzzlers

```
------ Task # 3 -------

reverseWords word = unwords( reverse( words word ) )

averageWordLength word = fromIntegral (l - numOfSpaces)  / fromIntegral numOfWords
 where l = length word
       numOfWords = length ( words word)
       numOfSpaces = numOfWords - 1
```

```
>>> averageWordLength "appa and baby yoda are the best"
3.5714285714285716
>>> averageWordLength "want me some coffee"
4.0
Ok, one module loaded.
>>> reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
>>> reverseWords "want me some coffee"
"coffee some me want"
```

Task 4 - Recursive List Processors

```
------ Task # 4 -------

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

isPalendrome [] = True
isPalendrome ( x : xs ) =  if ( length ( x : xs) ) == 1 then True
else ( if ( x == last xs ) then ( isPalendrome (head xs : drop 1
(init xs) ) ) else False )

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

>>> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>> list2set "need more coffee"
"ndmr cofe"
>>> isPalendrome ["coffee", "latte","coffee"]
True
>>> isPalendrome "racecar"
True
>>> isPalendrome "racecars"
False
>>> 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

```
------ Task # 5 -------

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

count search list = length [ x | x <- list, x == search ]

freqTable list = zip (list2set list) [ count x list | x <- (list2set list) ]
```

```
>>> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>> 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)]
>>> count 'e' "need more coffee"
5
>>> count 4 [1,2,3,4,4,2,3,1,4,14]
3
>>> count 4 [1,2,3,4,4,2,3,
```

Task 6 - Higher Order Functions

```
------ Task # 6 -------

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

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

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

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

```
>>> 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 "hey"
1
>>> vowelCount "mouse"
3
>>> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
```

Task 7 - An Interesting Statistic: nPVI

Task 7a - Test data

```
------ Task # 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]
```

```
>>> 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**

```
------ Task # 7b -------
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**

```
------ Task # 7c -------

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**

```
------ Task # 7d -------

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**

```
------ Task # 7e -------

half :: Int -> Double
half n = ( fromIntegral n) / 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

```
------ Task # 7f -------

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

```
------ Task # 7g -------

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

```
------ Task # 7h -------

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

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

```
------ Task # 7i -------

nPVI :: [Int] -> Double
nPVI xs = normalizer xs * sum (pairwiseTerms xs )

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

```
>>> 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 'a' symbols
('a',"– –––")
>>> assoc 'l' symbols
('l',"– ––– – –")
>>> find 'k'
"––– – –––"
>>> find 'g'
"––– ––– –"
```

Subtask 8c

```
>>> droplast7 "extravagant"
"extr"
>>> addletter "a" "m"
"a   m"
>>> addword "palm" "tree"
"palm       tree"
>>> droplast3 "finals week"
"finals w"
```

Subtask 8d

```
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'a'
"_ ___"
>>> encodeletter 'c'
"___ _ ___ _"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "hey"
"_ _ _ _   _   ___ _ ___ ___"
>>> encodeword "none"
"___ _   ___ ___ ___   ___ _   _"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _
   ___ _ ___ _   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
>>> encodemessage "need more sleep"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _
   _ _ _   _ ___ _ _   _   _   _ ___ ___ _"
>>> encodemessage "need more time"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _
   ___   _ _   ___ ___   _"
```