Haskell Programming Assignment #1: Various Computations

ABSTRACT:

The purpose of this programming assignment is to provide the student with 8 tasks that enhance their understanding of functions, recursive list processing, list comprehensions, and higher order functions in Haskell. Below contains structured code and demonstrations using GHCi.

Task 1: Mindfully Mimicking the Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: 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

Definitions

```
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--- ha.hs contains some simple numeric function
--- definitions
--------------------------------------------------------------------------
-- Thing 1

squareArea sideLength = sideLength * sideLength

--------------------------------------------------------------------------
-- Thing 2

circleArea radCircle = pi * ( radCircle * radCircle )

--------------------------------------------------------------------------
-- Thing 3

blueAreaOfCube sideLength = 6 * ( blueSquare - whiteDot )
      where blueSquare = squareArea sideLength
            whiteDot = circleArea ( sideLength / 4 )

--------------------------------------------------------------------------
-- Thing 4

paintedCube1 cubeOrder = if ( cubeOrder == 1 ) then 0
else if ( cubeOrder == 2 ) then 0
else 6 * ( cubeOrder - 2 ) ^ 2

--------------------------------------------------------------------------
-- Thing 5

paintedCube2 cubeOrder = if ( cubeOrder == 1 ) then 0
else if ( cubeOrder == 2 ) then 0
else 12 * ( cubeOrder - 2 )
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :cd C:\Haskell
ghci> :load ha
[1 of 1] Compiling Main             ( ha.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>
```

Task 3 – Puzzlers

Definitions

```
-----------------------------------------------------------------------
-- Thing 6

reverseWords string = unwords rString
      where stringList = words string
            rString = reverse stringList

-----------------------------------------------------------------------
-- Thing 7

averageWordLength string = ( stringLength / numOfString )
      where stringList = words string
            numOfString = fromIntegral ( length stringList )
            stringLength = fromIntegral ( length string ) - ( numOfString - 1 )
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
>>> reverseWords "want me some coffee"
"coffee some me want"
>>>  averageWordLength "appa and baby yoda are the best"
3.5714285714285716
>>> averageWordLength "want me some coffee"
4.0
>>>
```

Task 4 - Recursive List Processors

Definitions

```
--------------------------------------------------------------------------
-- Thing 8

list2set [] = []
list2set (x:xs) = x : list2set (filter ( /= x ) xs )

--------------------------------------------------------------------------
-- Thing 9

isPalindrome [] = True
isPalindrome [_] = True
isPalindrome (x:xs)
  | x == last xs = isPalindrome ( init xs )
  | otherwise = False

--------------------------------------------------------------------------
-- Thing 10

collatz 1 = [1]
collatz val
  | even val = val : collatz ( val `div` 2 )
  | otherwise = val : collatz ( 3 * val + 1 )
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>> list2set "need more coffee"
"ned morcf"
>>> 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

Definitions

```
-------------------------------------------------------------------------
-- Thing 11

count obj lst = numObj
      where numObj = length [ x | x <- lst, x == obj ]

-------------------------------------------------------------------------
-- Thing 12

freqTable lst = table
      where set = list2set lst
            table = [ ( x, count x lst ) | x <- set ]
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci>  :set prompt ">>> "
>>> 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),('e',5),('d',1),(' ',2),('m',1),('o',2),('r',1),('c',1),('f',2)]
>>>  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)]
>>>
```

Task 6 - Higher Order Functions

Definitions

```
-------------------------------------------------------------------------
-- Thing 13

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

-------------------------------------------------------------------------
-- Thing 14

triangleSequence val = map tgl [1..val]

-------------------------------------------------------------------------
-- Thing 15

vowelCount string = length ( filter ( \x -> x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u' )
string )

-------------------------------------------------------------------------
-- Thing 16

lcsim f p lst = map f ( filter ( \x -> p x ) lst )
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> 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]
>>>
```

Task 7 - An Interesting Statistic: nPVI

Task 7a - Test data

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> a
[2,5,1,3]
>>> b
[1,3,6,2,5]
>>> c
[]
>>> 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

Definitions

```
-------------------------------------------------------------------
-- Thing 1

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

pairwiseValues lst = zip lst ( tail lst )
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> pairwiseValues a
[(2,5),(5,1),(1,3)]
>>> pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
>>> pairwiseValues c
[]
>>> 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

Definitions

```
-------------------------------------------------------------------
-- Thing 2

pairwiseDifferences :: [Int] -> [Int]

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> pairwiseDifferences a
[-3,4,-2]
>>> pairwiseDifferences b
[-2,-3,4,-3]
>>> pairwiseDifferences c
[]
>>> 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

Definitions

```
-----------------------------------------------------------------------
-- Thing 3

pairwiseSums :: [Int] -> [Int]

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> pairwiseSums a
[7,6,4]
>>> pairwiseSums b
[4,9,8,7]
>>> pairwiseSums c
[]
>>> 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

Definitions

```
--------------------------------------------------------------------
-- Thing 4

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

pairwiseHalves :: [Int] -> [Double]
pairwiseHalves lst = map ( \(x) -> half x ) lst
```

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> 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

Definitions

```
--------------------------------------------------------------------
-- Thing 5

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> pairwiseHalfSums a
[3.5,3.0,2.0]
>>> pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
>>> pairwiseHalfSums c
[]
>>> 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

Definitions

```
-----------------------------------------------------------------------
-- Thing 6

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> 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
[]
>>> 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

Definitions

```
-----------------------------------------------------------------------
-- Thing 7

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

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
>>> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
>>> pairwiseTerms c
[]
>>> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
>>> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555555555
55556,1.6]
>>>
```

Task 7i - The nPVI function

Definitions

```
-----------------------------------------------------------------------
-- Thing 8

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

Demo

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load npvi
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> nPVI a
106.34920634920636
>>> nPVI b
88.09523809523809
>>> nPVI c
-0.0
>>> nPVI u
0.0
>>> nPVI x
124.98316498316497
>>>
```

Task 8 - Historic Code: The Dit Dah Code

Subtask 8a

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ditdah
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> 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

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ditdah
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> assoc 's' symbols
('s',"- - -")
>>> assoc 'l' symbols
('l',"- --- - -")
>>> find 'h'
"- - - -"
>>> find 'o'
"--- --- ---"
>>>
```

Subtask 8c

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ditdah
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> addletter "s" "e"
"s   e"
>>> addword "water" "bottle"
"water       bottle"
>>> droplast3 "computer science"
"computer scie"
>>> droplast7 "computer science"
"computer "
>>>
```

Subtask 8d

```
PS C:\Users\e_rro> ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :c C:\Haskell
ghci> :load ditdah
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'f'
"_ _ ___ _"
>>> encodeletter 'j'
"_ ___ ___ ___"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "wow"
"_ ___ ___   ___ ___ ___   _ ___ ___"
>>> encodeword "nice"
"___ _   _ _   ___ _ ___ _   _"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _       ___ _
 ___ _   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
```

```
>>> encodemessage "i am hungry"
"_ _       _ ___   ___ ___       _ _ _ _   _ _ ___   ___ _   ___ ___ _   _ ___
 _   ___ _ ___ ___"
>>> encodemessage "how are you"
"_ _ _ _   ___ ___ ___   _ ___ ___       _ ___   _ ___ _   _       ___ _ ___ _
__   ___ ___ ___   _ _ ___"
>>>
```