Haskell Assignment: Various Computations

Learning Abstract

In this programming assignment, I will be looking at the ins and outs of the programming language Haskell to get acquainted with it and better appreciate its syntax as a functional language.

Task 1: Mindfully Mimicking the Demo

```
Last login: Thu May  4 15:16:37 on ttys000
mhanheinkhant@Mhans-MacBook-Air ~ % ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> length [2,3,5,7]
4
ghci> words "need more coffee"
["need","more","coffee"]
ghci> unwords["need","more","coffee"]
"need more coffee"
ghci> reverse "need more coffee"
"eeffoc erom deen"
ghci> reverse ["need","more","coffee"]
["coffee","more","need"]
ghci> head ["need","more","coffee"]
"need"
ghci> tail ["need","more","coffee"]
["more","coffee"]
ghci> last ["need","more","coffee"]
"coffee"
ghci> init ["need","more","coffee"]
["need","more"]
ghci> take 7 "need more coffee"
"need mo"
ghci> drop 7 "need more coffee"
"re coffee"
ghci> ( \x -> length x > 5 ) "Friday"
True
ghci> ( \x -> length x > 5 ) "uhoh"
False
ghci> ( \x -> x /= ' ' ) 'Q'
True
ghci> ( \x -> x /= ' ' ) ' '
False
ghci> filter ( \x -> x /= ' ' ) "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
ghci> :quit
Leaving GHCi.
mhanheinkhant@Mhans-MacBook-Air ~ %
```

Task 2: Numeric Function Definitions

Code

```
squareArea sideLength = sideLength * sideLength

circleArea radius = pi * (radius * radius)

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

paintedCube1 1 = 0
paintedCube1 n = 6 * (n - 2) * (n - 2)

paintedCube2 1 = 0
paintedCube2 m = 12 * (m - 2)
```

Demo

```
Last login: Thu May  4 16:29:48 on ttys000
mhanheinkhant@Mhans-MacBook-Air ~ % ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ha.hs
[1 of 1] Compiling Main             ( Desktop/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.
mhanheinkhant@Mhans-MacBook-Air ~ %
```

Task 3: Puzzlers

Code

```
reverseWords lst = unwords (reverse (words lst))

averageWordLength :: String -> Double
averageWordLength lst2 = fromIntegral(length(filter ( \x -> x /= ' ' )
lst2)) / fromIntegral (length(words lst2))
```

Demo

```
Last login: Thu May  4 23:12:12 on ttys002
mhanheinkhant@res-dhcp-129-3-136-78 ~ % ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ha2.hs
[1 of 1] Compiling Main             ( Desktop/ha2.hs, interpreted )
Ok, one module loaded.
>>> reverseWords "appa abd baby yoda are the best"
"best the are yoda baby abd appa"
>>> reverseWords "want me some coffee"
"coffee some me want"
>>> reverseWords "banana dog box thousand"
"thousand box dog banana"
>>> reverseWords "I went outside and drank water"
"water drank and outside went I"
>>> averageWordLength "appa abd baby yoda are the best"
3.5714285714285716
>>> averageWordLength "want me some coffee"
4.0
>>> averageWordLength "banana dog box thousand"
5.0
>>> averageWordLength "I went outside and drank water"
4.166666666666667
>>> :quit
Leaving GHCi.
mhanheinkhant@res-dhcp-129-3-136-78 ~ %
```

Task 4: Recursive List Processors

Code

```
list2set [] = []
list2set (n:m) = n : list2set (filter (/= n) m)

isPalindrome [] = True
isPalindrome [_] = True
isPalindrome (n:m)
 | n == last m = isPalindrome (init m)
 | otherwise = False

collatz :: Integer -> [Integer]
collatz 0 = [0]
collatz 1 = [1]
collatz n
  | n `mod` 2 == 0 = n : collatz (n `div` 2)
  | otherwise = n : collatz (3 * n + 1)
```

Demo

```
Last login: Sun May  7 05:20:52 on ttys000
mhanheinkhant@res-dhcp-129-3-136-93 ~ % ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ha3.hs
[1 of 1] Compiling Main             ( Desktop/ha3.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"
"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]
>>> :quit
Leaving GHCi.
mhanheinkhant@res-dhcp-129-3-136-93 ~ %
```

Task 5: List Comprehensions

Code

```
list2set [] = []
list2set (n:m) = n : list2set (filter (/= n) m)

count object list = timesPresent
      where timesPresent = length [ x | x <- list, x == object ]


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

Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ha4.hs
[1 of 1] Compiling Main             ( Desktop/ha4.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 'f' "funky flying car on the turf"
3
>>> count 8 [1,9,8,8,5,3,65,8,5,3,8,9,8]
5
>>> 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)]
>>> freqTable "funky flying car on the turf"
[('f',3),('u',2),('n',3),('k',1),('y',2),(' ',5),('l',1),('i',1),('g',1),('c',1),('a',1),('r',2),
('o',1),('t',2),('h',1),('e',1)]
>>> freqTable [1,9,8,8,5,3,65,8,5,3,8,9,8]
[(1,1),(9,2),(8,5),(5,2),(3,2),(65,1)]
>>>
```

Task 6: Higher Order Functions

Code

```
tgl 1 = 1
tgl n = n + tgl (n-1)

triangleSequence n = map tgl [1..n]

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

lcsim fun pre lst = [ fun n | n <- lst, pre n ]
```

Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ha5.hs
[1 of 1] Compiling Main             ( Desktop/ha5.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.
mhanheinkhant@res-dhcp-129-3-137-157 ~ %
```

Task 7: An Interesting Statistic: nPVI

Task 7a – Test data

Code

```
-- Test data
a :: [Int]
a = [2,5,1,3]

b :: [Int]
b = [1,3,6,2,5]

c :: [Int]
c = [3,6,2,7,9,3,5,7,6,2]

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]
```

Task 7b – The pairwiseValues function

Code

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

Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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)]
>>>
```

Task 7c – The pairwiseDifferences functions

Code

```
pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences lst = map (\(n,m) -> n - m) (pairwiseValues lst)
```

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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]
>>>
```

Task 7d – The pairwiseSums functions

Code

```
pairwiseSums :: [Int] -> [Int]
pairwiseSums lst = map (\(n,m) -> n + m) (pairwiseValues lst)
```

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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]
>>>
```

Task 7e – The pairwiseHalves functions

Code

```
half :: Int -> Double
half number = (fromIntegral number) / 2
pairwiseHalves :: [Int] -> [Double]
pairwiseHalves lst = map half lst
```

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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]
>>>
```

Task 7f – The pairwiseHalfSums function

Code

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

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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]
>>>
```

Task 7g – The pairwiseTermPairs function

Code

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

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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)]
>>>
```

Task 7h – The pairwiseTerms function

Code

```
term :: (Int,Double) -> Double
term ndPair = abs (fromIntegral (index1 ndPair) / (index2 ndPair))
pairwiseTerms :: [Int] -> [Double]
pairwiseTerms lst = map term (pairwiseTermPairs lst)
```

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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]
>>>
```

Task 7i – The nPVI function

Code

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

Demo

```
>>> :load Desktop/npvi.hs
[1 of 1] Compiling Main             ( Desktop/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
>>>
```

Task 8: Historic Code: The Dit Dah Code

Subtask 8a

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load Desktop/ditdah.hs
[1 of 1] Compiling Main             ( Desktop/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 'b' symbols
('b',"--- - - -")
>>> assoc 'o' symbols
('o',"--- --- ---")
>>> find 'q'
"--- --- - ---"
>>> find 'h'
"- - - -"
>>>
```

Subtask 8c

```
>>> addletter "b" "h"
"b   h"
>>> addword "wake" "up"
"wake       up"
>>> droplast3 "right now"
"right "
>>> droplast7 "right now"
"ri"
>>>
```

Subtask 8d

```
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'g'
"___ ___ _"
>>> encodeletter 't'
"___"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "genius"
"___ ___ _   _   ___ _   _ _   _ _ ___   _ _ _"
>>> encodeword "turing"
"___   _ _ ___   _ ___ _   _ _   ___ _   ___ ___ _"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _       ___ _ _
__ _   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
>>> encodemessage "enigma code broken"
"_   ___ _   _ _   ___ ___ _   ___ ___   _ ___       ___ _ ___ _   ___ ___ ___
 ___ _ _   _       ___ _ _ _   _ ___ _   ___ ___ ___   ___ _ ___   _   ___ _"
>>> encodemessage "summer is here"
"_ _ _   _ _ ___   ___ ___   ___ ___   _   _ ___ _       _ _   _ _ _       _ _ _
 _   _   _ ___ _   _"
>>>
```