# Haskell Programming Assignment: Various Computations

## Learning Abstract

In this assignment we learn about functions, recursive list processing, list comprehensions, and higher order functions in Haskell.

### Task 1 - Mindfully Mimicking the Demo

```
 PS C:\ghcup\test> 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?"
>>> :quit
```

## Task 2 - Numeric Function Definitions

### Code

```
squareArea a = a * a
-----------------------------------------------
circleArea a = pi * (squareArea a)
-----------------------------------------------
blueAreaOfCube l = a - b
   where a = 6 * (squareArea l)
         b = 6 * (circleArea (l / 4))
-----------------------------------------------

paintedCube1 a = if (a == 1) then 0
else if (a == 2 ) then 0
else 6 * (a - 2) ^ 2
-----------------------------------------------
paintedCube2 a = if (a == 1) then 0
else if (a == 2) then 0
else 12 * (a - 2)
```

### Demo

```
>>> :load "task2.hs"
[1 of 1] Compiling Main             ( task2.hs, interpreted )
Ok, one module loaded.
>>> squareArea 10
100
>>> squareArea 12
144
>>> circleArea 10
314.1592653589793
>>> circleArea 12
452.3893421169302
>>> blueAreaOfCube 12
694.3539967061512
>>> blueAreaOfCube 10
482.19027549038276
```

```
>>> 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]
>>>
```

## Task 3 - Puzzlers

### Code

```
reverseWords word = unwords (reverse (words word))
averageWordLength word = ( stringLength / wordLength )
   where wordLength = fromIntegral (length (words word))
         stringLength = fromIntegral ( length word ) - (wordLength - 1)
```

### Demo

```
>>> :load "task3.hs"
[1 of 1] Compiling Main             ( task3.hs, interpreted )
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"
>>> averageWordLength "appa and baby yoda are the best"
3.5714285714285716
```

```
>>> averageWordLength "want me some coffee"
4.0
>>> :quit
```

## Task 4 - Recursive List Processors

### Code

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


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

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

### Demo

```
>>> :load "task4.hs"
[1 of 1] Compiling Main             ( task4.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
```

## Task 5 - List Comprehensions

### Code

```
count letter word = length [x | x <- word, x == letter]

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

freqTable word = answer
   where list = list2set word
         answer = [(a, count a word)| a <- word]
```

### Demo

```
>>> :load "task5.hs"
[1 of 1] Compiling Main             ( task5.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 'h' "happy birthday to you"
2
>>> count 2 [1,2,3,4,5,2,3,1,2,3]
3
>>> freqTable "need more coffee"
[('n',1),('e',5),('e',5),('d',1),(' ',2),('m',1),('o',2),('r',1),('e',5),(' ',2),('c',1),('o',2),('f',2),('f',2),('e',5),('e',5)]
>>> freqTable [1,2,3,2,3,4,3,4,5,4,5,6]
[(1,1),(2,2),(3,3),(2,2),(3,3),(4,3),(3,3),(4,3),(5,2),(4,3),(5,2),(6,1)]
>>> freqTable "happy birthday to you"
[('h',2),('a',2),('p',2),('p',2),('y',3),(' ',3),('b',1),('i',1),('r',1),('t',2),('h',2),('d',1),('a',2),('y',3),(' ',3),('t',2),('o',2),('
  ',3),('y',3),('o',2),('u',1)]
>>> freqTable [1,2,3,4,5,2,3,1,2,3]
[(1,2),(2,3),(3,3),(4,1),(5,1),(2,3),(3,3),(1,2),(2,3),(3,3)]
>>> :quit
```

## Task 6 - Higher Order Functions

### Code

```
tgl int = foldl (+) 0 [1..int]

triangleSequence int = map tgl [1..int]

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

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

### Demo

```
>>> tgl 5
15
>>> tgl 10
55
>>> tgl 20
210
>>> tgl 35
630
>>> 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 5
[1,3,6,10,15]
>>> triangleSequence 35
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210,231,253,276,300,325,351,378,406,435,4
 65,496,528,561,595,630]
>>> vowelCount "cat"
1
>>> vowelCount "mouse"
3
>>> vowelCount "constraint"
3
>>> vowelCount "happy"
1
>>> 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]
>>> lcsim tgl even [1..10]
[3,10,21,36,55]
>>> lcsim length (\x -> elem (last x) "aeiou") animals
[]
```

## 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 = [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

### Code

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

### 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

### Code

```
pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences list = map (\(x,y) -> x - y ) (pairwiseValues list)
```

### 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

### Code

```
pairwiseSums :: [Int] -> [Int]
pairwiseSums list = map ( \(x,y) -> x + y) (pairwiseValues list)
```

### Demo

```
>>> :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]
>>>
```

## Task 7e - The pairwiseHalves function

### Code

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

### Demo

```
>>> 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 list = pairwiseHalves ( pairwiseSums list)
```

### Demo

```
>>> :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]
>>>
```

### Task 7g - The pairwiseTermPairs function

### Code

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

### Demo

```
>>> :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)]
>>>
```

### Task 7h - The pairwiseTerms function

### Code

```
term :: (Int,Double) -> Double
term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )
pairwiseTerms :: [Int] -> [Double]
pairwiseTerms list = map term (pairwiseTermPairs list)
```

### Demo

```
>>> :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
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6
 666666666666666]
>>> 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 xs = normalizer xs * sum ( pairwiseTerms xs )
   where normalizer xs = 100 / fromIntegral ( ( length xs ) - 1 )
```

### Demo

```
>>> :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
>>>
```

## Task 8 - Historic Code: The Dit Dah Code

### SubTask 8a

```
>>> :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',"--- --- - -")]
```

### Subtask 8b

```
('a',"- ---")
>>> assoc 'z' symbols
('z',"--- --- - -")
>>> find 'a'
"- ---"
>>> find 'z'
"--- --- - -"
>>>
```

### Subtask 8c

```
>>> addletter "a" "t"
```

```
"a   t"
>>> addword "happy" "birthday"
"happy       birthday"
>>> droplast3 "sakshyam"
"saksh"
>>> droplast7 "sakshyam"
"S"
```

### Subtask 8d

```
>>> encodeletter 'm'
"--- ---"
>>> encodeletter 'z'
"--- --- - -"
>>> encodeletter 's'
"- - -"
>>> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
>>> encodeword "happy"
"- - - -   - ---   - --- --- -   - --- --- -   --- - --- ---"
>>> encodeword "good"
"--- --- -   --- --- ---   --- --- ---   --- - -"
>>> encodemessage "need more coffee"
"--- -   -   -   --- - -     --- ---   --- --- ---   - --- -   -     --- - --- -   --- --- ---   - - --- -   - - --- -   -   -"
>>> encodemessage "happy birthday"
"- - - -   - ---   - --- --- -   - --- --- -   --- - --- ---     --- - - -   - -   - --- -   ---   - - - -   --- - -   - ---   --- - --- ---"
>>> encodemessage "good night"
"--- --- -   --- --- ---   --- --- ---   --- - -     --- -   - -   --- --- -   - - - -   ---"
>>>
```