First Haskell Programming Assignment

Learning Abstract - In this assignment, I learned how to program in Haskell. In many ways its similar to Prolog

Task 1

```
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.
```

Task 2

```
--squareArea : Calulates the area of a square
1   --squareArea : Calulates the area of a square
2   squareArea num = num * num
3
4   -- circleArea : Calculates the area of a cicrle
5   CircleArea radius = radius * radius * pi
6
7   -- blueAreaofCube: Calculates the blue area of a cube when 1/4 of each
8   --face has a white dot
9   blueAreaOfCube edge = 6 * ((squareArea edge) - (circleArea (edge/4)))
10
11  -- paintedCube1 : calculates the number of cubes with only one painted
12  --face
13  paintedCube1 order = if (order > 2) then ( 6 *((order - 2) ^ 2)) else 0
14
15  -- paintedCube2 : Calculates the number of cubes with two painted faces
16  paintedCube2 order = if (order > 2 ) then ( 6* (2 *(order - 2 ))) else 0
```

```
>>> 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 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

-- reverse words : function that takes ina list of

```
1  -- reverse words : function that takes ina list of words and reverses
2  --the order of those words
3  reverseWords w = unwords (reverse (words w))
4
5  -- averageWordLength: function that takes a list of words and calculates -- the average word length
6  averageWordLength n = fromIntergral (strLength - numOfSpaces ) / fromIntergral numOfWords
7      where strLength = length n
8      numOfWords = length (words n)
9      numOfSpaces = nunOfSpaces - 1
```

```
>>> 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

-- list2set : takes in a list of objects, and retu

```
1   -- list2set : takes in a list of objects, and returns the list with no
2   --duplicates
3   list [] = []
4   list2set (x:xs) = if (element x xs ) then (function) else (x : function)
5       where function = list2setset xs
6
7   --isPalindrome: takes in a list of objects and returns a boolean based
8   --on whether the list is palondrome of not
9   isPalidrome [] = True
10  isPalidrome (x:xs) =
11      if (length(x:xs)) == 1 then True
12      else (if(x== last xs ) then (f) else False )
13      where f = is Palidrome (head xs: drop 1 (init xs))
```

```
>>> 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
```

Task 5

```haskell
-- count
count n xs = length[x|x<- xs, x==n]

-- freqTable
freqTable xs = [(x, y) | x<-list2set xs, y<-[count s xs|s<-[x]]]
```

```
>>> 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

```haskell
-- tgl
tgl :: Int -> Int
tgl n = foldl (+) 0 [1..n]

-- triangleSequence
triangleSequence :: Int -> [Int]
triangleSequence n = map tgl [1..n]

--vowelCount
vowelCount :: String -> Int
vowelCount n = length $ filter (\x -> x 'elem' "aeiou") n

-- lcsim
lcsim m f e = map(m)(filter(f)e)
```

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

7a

```
-- 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]
```

7b / c / d

```
-- pairWiseValues
1  -- pairWiseValues
2  pairwiseValues :: [Int] -> [(Int, Int)]
3  pairwiseValues xs = zip xs $ tail xs
4
5  pairwiseDifferences :: [Int] -> [Int]
6  pairwiseDifferences xs = map (\(x,y) ->  x - y ) $ pairwiseValues xs
7
8  pairwiseSums :: [Int] -> [Int]
9  pairwiseSum xs = map (\(x,y) ->  x + y ) $ pairwiseValues 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)]
>>> 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)]
```

```
>>> pairwiseDifferences a
[-3,4,-2]
>>> pairwiseDifferences b
[-2,-3,4,-3]
>>> pairwiseDifferences c
[0,2,1,0,-1,0,-2,0]
>>> pairwiseDifferences u
[0,0,0,0,0,0,0,0]
>>> pairwiseDifferences x
[-8,7,-6,5,-4,5,-6,7,-8]
>>>
```

```
>>> pairwiseSums a
[7,6,4]
>>> pairwiseSums b
[4,9,8,7]
>>> pairwiseSums c
[8,6,3,2,3,4,6,8]
>>> pairwiseSums u
[4,4,4,4,4,4,4,4,4]
>>> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
>>>
```

7e

```
1  half :: Int -> Double
2  half number = (fromIntergral num) / 2
3  pairwiseHalves :: [Int] -> [Double ]
4  pairwiseHalves xs = map half xs
```

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

7f

```
pairwiseHalfSums :: [Int] -> [Double]
pairwiseHalfSums xs = pairwiseHalves $ pairwiseSums xs
```

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

7g

```
pairwiseTermPairs :: [Int] -> [(Int,Double)]
pairwiseTermPairs xs = zip (pairiseDifferences xs )(pairwiseHalfSums xs)
```

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

7h

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

7i

```
>>> nPVI a
106.34920634920636
>>> nPVI b
88.09523809523809
>>> nPVI c
33.33333333333333
>>> nPVI u
0.0
>>> nPVI x
124.98316498316497
>>>
```

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

8b

```
>>> assoc 'x' symbols
('x',"--- - - ---")
>>> assoc 'y' symbols
('y',"--- - --- ---")
>>> find 'w'
"- --- ---"
>>> find 'z'
"--- --- - -"
>>>
```

8c

```
>>> addletter "x" "-- - -"
"x   -- - -"
>>> addword "good" "----- -- --"
"good       ----- -- --"
>>> droplast3 "good"
"g"
>>> droplast7 "this is a test"
"this is"
```

8d

```
>>> encodeletter 'm'
"--- ---"
>>> encodeletter 'x'
"--- - - ---"
>>> encodeletter 'y'
"--- - --- ---"
>>> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
>>> encodeword "nay"
"--- -   - ---   --- - --- ---"
>>> encodeword "test"
"---   -   - - -   ---"
>>> encodemessage "need more coffee"
"--- -   -   -   --- - -       --- ---   --- --- ---   - --- -   -       --- - --- -   --- --- ---   - - --- -   - - ---
 -   -   -"
>>> encodemessage "secret message here"
"- - -   -   --- - --- -   - --- -   -   ---       --- ---   -   - - -   - - -   - ---   --- --- -   -       - - - -   -
   - --- -   -"
>>> encodemessage "impossible to crack"
"- -   --- ---   - --- --- -   --- --- ---   - - -   - - -   - -   --- - - -   - --- - -   -       ---   --- --- ---
   --- - --- -   - --- -   - ---   --- - --- -   --- - ---"
```