

Title: Haskell Assignment #1: Various Computations

Abstract:

Intro to Haskell

Task 1:

```
>>> :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:

squareArea side = side * side

circleArea radius = pi * radius * radius

blueAreaOfCube side = 6 * ((side*side) - (pi * (side/4) * (side/4)))

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

paintedCube2 n = if n<2 then 0 else 12 * (n-2)

```
> 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]
```

Task 3:

```
reverseWords string = unwords(reverse(words(string)))
averageWordLength string = fromIntegral(sum(map length list)) /
fromIntegral(denominator)
  where list = words(string)
        denominator = length list
```

```
> 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
>
> reverseWords "ooga booga"
"booga ooga"
> reverseWords "my name is tim"
"tim is name my"
> averageWordLength "ooga booga"
4.5
> averageWordsLength "my name is tim"

<interactive>:11:1: error:
    • Variable not in scope: averageWordsLength :: [Char] -> t
    • Perhaps you meant 'averageWordLength' (line 8)
> averageWordLength "my name is tim"
2.75
> █
```

Task 4:

list2set list = if (length list) == 0 then [] else (if elem (head list) tsil then tsil
else (head list) : tsil)

where tsil = list2set(tail list)

denominator = length list

isPalindrome list = if (length list) < 2 then True else (if head list == last list
then isPalindrome (init (tail list)) else False)

collatz n = if n == 1 then [1] else (n : list)

where nn = if (rem n 2) == 0 then (div n 2) else (3 * n + 1)

list = (collatz nn)

```
* list2set[1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
* list2set "need more coffee"
"ndmr cofe"
* 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:

```
count obj list = if(length list == 0) then 0 else (if head list == obj then 1 +
count obj (tail list) else count obj (tail list))
freqTable list = [ (x, count x list) | x <- set]
  where set = list2set list
```

```
> freqTable [1,2,4,5,3,5,3,1]
[(2,1),(4,1),(5,2),(3,2),(1,2)]
> 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),('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 'a' "salamanca"
4
> count ' ' "asfr ser qwuy ois zser rewtg gi kol lra  "
12
```

```
> freqTable ["Timofej", "Sergejevicz", "Germakovskij", "Ivan", "Sergejevicz", "Germakovskij", "Jonafan", "Andrejevicz", "Germakovskij"]
[("Timofej",1),("Ivan",1),("Sergejevicz",2),("Jonafan",1),("Andrejevicz",1),("Germakovskij",3)]
> freqTable [1,2,2,2,3,6,3,1,8,7,6,4,3,2]
[(1,2),(8,1),(7,1),(6,2),(4,1),(3,3),(2,4)]
>
```

Task 6:

```
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)
```

```

> 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
> tgl 44
990
> tgl 69
2415
> triangleSequence 4
[1,3,6,10]
> triangleSequence 8
[1,3,6,10,15,21,28,36]

> vowelCount "aooga"
5
> vowelCount "hrrrgnnjkhtlf"
0

```

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:

```

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

```

```

> 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)]
>

```

<interactive>:7:1: error:

- Variable not in scope: `paitwiseValues :: [Int] -> t`
- Perhaps you meant `'pairwiseValues'` (line 35)

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

`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

`half :: Int -> Double`

`half number = ( fromIntegral number ) / 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]
> pairwiseHalves x
[0.5,4.5,1.0,4.0,1.5,3.5,1.0,4.0,0.5,4.5]
> 

```


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

```
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

term :: (Int,Double) -> Double

term ndPair = abs (fromIntegral (fst ndPair) / (snd ndPair))

pairwiseTerms :: [Int] -> [Double]

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]
> pairwiseTerms u
[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

nPVI :: [Int] -> Double

nPVI xs = normalizer xs * sum (pairwiseTerms xs)

where 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 8a

```
➤ dit
"_"
➤ dah
"___"
➤ (+++) dah dit
"___ _"
➤ 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', "___ ___ _ _")]
➤
```

Task 8b

```
➤ assoc 'w' symbols
('w', "_ _ ___ ___")
➤ assoc 'j' symbols
('j', "_ _ ___ ___ ___")
➤ find 'c'
"___ _ ___ _"
➤ find 'y'
"___ _ ___ ___"
```

Task 8c

```
❖ addletter "c" "z"
"c  z"
❖ addword "cziki" "briki"
"cziki      briki"

❖ droplast3 "bipoc"
"bi"
❖ droplast7 "bifurcate"
"bi"
```

Task 8d

```
❖ encodeletter 'm'
"__ _"
❖ encodeletter 'h'
"_ _ _"
❖ encodeletter 'z'
"__ _ _ _"
❖ encodeword "yay"
"__ _ _ _ _ _ _ _ _ _"
❖ encodeword "timofej"
"__ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
❖ encodeword "tymofiy"
"__ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
❖ encodeword "need more coffee"
"__ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
❖ encodeword "someone pls call an ambulance"
"__ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
❖ encodeword "jk nvm"
"__ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
```