Haskell Programming Assignment: Various Computations

Task 1

Demo

```
Last login: Thu May  4 14:55:13 on ttys001
skye@Skyes-Mac ~ % ghci
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> length[2,3,4,5,7]
5
ghci> :quit
Leaving GHCi.
skye@Skyes-Mac ~ % 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'

<interactive>:14:13: error:
    Parser error on `''`
    Character literals may not be empty
ghci> ( \x ->x /= ' ' ) 'Q'
True
ghci> ( \x -> x /= ' ') ' "

<interactive>:16:22: error:
    lexical error in string/character literal at end of input
ghci> ( \x -> x /= ' ') ' '
False
ghci> filter( \x -> x /= ' ') "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
ghci> :quit
Leaving GHCi.
skye@Skyes-Mac ~ %
```

Task 2 - Numeric Function Definitions

Task 2(1)

```
--- Function 1
squareArea x = x * x

--- Function 2
circleArea x = 3.141592653589793238 * ( x * x )

--- Function 3
blueAreaOfCube y = s * 6
   where s = squareArea y - circleArea x1
         x1 = y / 4


--- Function 4
paintedCube1 c =
     if ( c < 3 ) then 0 else
     count * 6
     where count = ( c - 2 ) * ( c - 2 )


--- Function 5
paintedCube2 c =
     if ( c < 3 ) then 0 else
     count * 12
     where count = ( c - 2 )
```

Demo

```
ghci> :load Desktop/ha1.hs
[1 of 1] Compiling Main             ( Desktop/ha1.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> :set prompt ">>> "
>>> 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 paintedCube [1..10]

<interactive>:11:5: error:
    • Variable not in scope: paintedCube :: a0 -> b
    • Perhaps you meant one of these:
        ‘paintedCube1’ (line 15), ‘paintedCube2’ (line 22)
>>> map paintedCube2 [1..10]
[0,0,12,24,36,48,60,72,84,96]
>>>
```

Task 3 - Puzzlers

Task 3 (a)

```
--- Task 3 - Puzzlers
--- Function 1
reverseWords sc = unwords ( reverse ( words sc ) )


--- Function 2
averageWordLength sc =
   let wordLst = words sc
       totalLen = fromIntegral ( sum (map length wordLst ) )
       numWords = fromIntegral ( length wordLst )
    in if numWords == 0
          then 0.0
       else totalLen / numWords
```

Task 3 (b)

```
[>>> :load Desktop/ha1.hs
 [1 of 1] Compiling Main             ( Desktop/ha1.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
 Leaving GHCi.
```

Task 4 - Recursive List Processors

Code

```
--- Task 4 - Recursive List Processors
--- Function 1
list2set [] = []
list2set (x:xs)
    | elem x xs = list2set xs
    | otherwise = x : list2set xs


--- Function 2
isPalindrome [] = True
isPalindrome [_] = True
isPalindrome (x:xs)
  | x == last xs = isPalindrome (init xs)
  | otherwise = False


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

Demo

```
>>> :load Desktop/ha1.hs
[1 of 1] Compiling Main             ( Desktop/ha1.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"
"ndmr cofe"
>>> isPalindrome ["coffee", "latte", "coffee"]
True
>>> isPalindrome ["coffee", "latte", "espresso", "coffee"]
False
>>> isPlaindrome [1,2,5,7,11,13,11,7,5,3,2]

<interactive>:9:1: error:
    • Variable not in scope: isPlaindrome :: [a0] -> t
    • Perhaps you meant ‘isPalindrome’ (line 51)
>>> isPalindrome [1,2,5,7,11,13,11,7,5,3,2]
False
>>> isPalindrome [2,3,5,7,11,13,7,5,3,2]
False
>>> 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

Code

```
--- Task 5
--- Function 1
count obj lst = length [ x | x <- lst, x == obj ]
```

```
--- Function 2

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

Demo

```
>>> :load Desktop/ha1.hs
[1 of 1] Compiling Main             ( Desktop/ha1.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
>>> 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)]
>>>
```

Task 6 - Higher Order Functions

Code

```
--- Task 6-Higher Order Functions
--- Function 1
tgl 1 = 1
tgl x = x + tgl (x-1)

--- Function 2
triangleSequence x = map tgl [1..x]


--- Function 3
vowelCount sofl = length ( filter (\x -> elem x "aeiou") sofl)
```

```
--- Function 4
lcsim fun pre lst = [ fun x | x <- lst, pre x ]
```

Demo

```
>>> :load Desktop/ha1.hs
[1 of 1] Compiling Main             ( Desktop/ha1.hs, interpreted )
Ok, one module loaded.
>>> tgl 2
3
>>> 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 "mouse"
3
>>> lcism tgl odd [1..15]

<interactive>:38:1: error:
    • Variable not in scope:
        lcism :: (t0 -> t0) -> (a0 -> Bool) -> [a1] -> t
    • Perhaps you meant ‘lcsim’ (line 91)
>>> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
>>> animals = ["element", "lion", "tiger", "orangatan", "jaguar"]
>>> lcsim length (\w -> elem (head w) "aeiou") animals
[7,9]
>>>
```

Task 7 - An Interesting Statistic: nPVI

7a

```
-- 7a
-- 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]
```

7b

```
--- 7b - The pairwiseValue function
pairwiseValues :: [Int] -> [(Int,Int)]
pairwiseValues lst = zip lst (tail lst)
```

```
>>> :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)]
>>> pairwiseVlaues b

<interactive>:35:1: error:
    • Variable not in scope: pairwiseVlaues :: [Int] -> t
    • Perhaps you meant one of these:
        'pairwiseValues' (line 21), 'pairwiseHalves' (line 36),
        'pairwiseSums' (line 29)
>>> 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)]
>>>
```

7c

```
--- 7c The pairwiseDifferences
pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences lst = map (\(x,y) -> x - y ) (
pairwiseValues lst )
```

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

7d

```
--- 7d The pairwiseSums
pairwiseSums :: [Int] -> [Int]
pairwiseSums lst = map (\(x,y) -> x + y ) ( pairwiseValues lst )
```

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

7e

```
--- 7e The pairwiseHalves
half :: Int -> Double
half number = ( fromIntegral number ) / 2
pairwiseHalves :: [Int] -> [Double]
pairwiseHalves lst = map half lst
```

```
>>> 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

```
--- 7f The pairwiseHalfSums
pairwiseHalfSums :: [Int] -> [Double]
pairwiseHalfSums lst = pairwiseHalves ( pairwiseSums lst )
```

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

7g

```
--- 7g The pairwiseTermPairs
pairwiseTermPairs :: [Int] -> [(Int,Double)]
pairwiseTermPairs lst = zip ( pairwiseDifferences lst )
(pairwiseHalfSums lst )
```

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

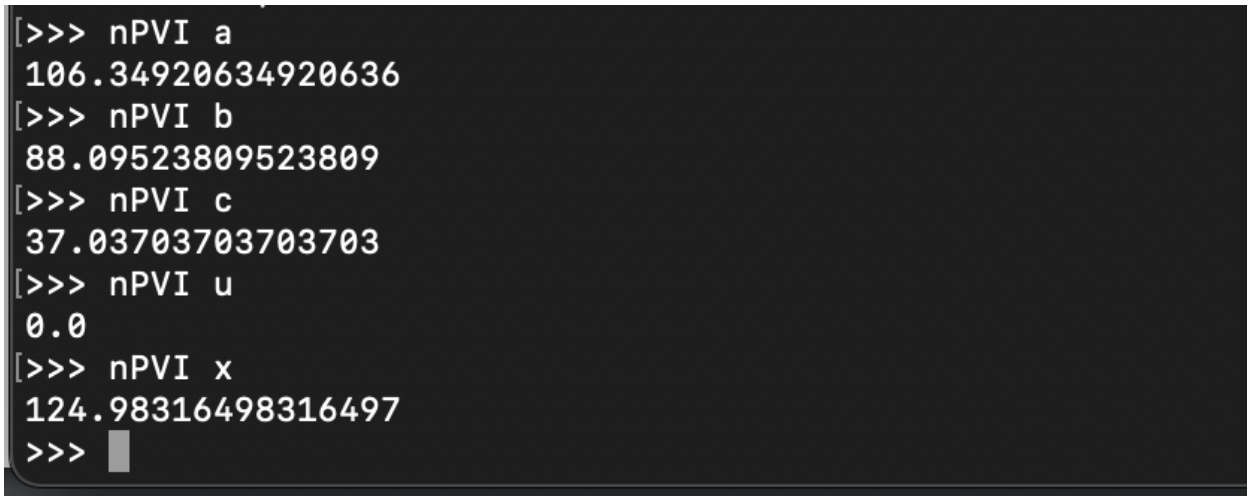
7h

```
--- 7h The pairwiseTerms
term :: (Int,Double) -> Double
term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair )
)
pairwiseTerms :: [Int] -> [Double]
pairwiseTerms lst = map term ( pairwiseTermPairs lst )
```

```
>>> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
>>> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
>>> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.66666666
6666666,0.0,0.6666666666666666]
>>> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
>>> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.55555
55555555556,1.6]
>>>
```

7i

```
--- 7i The nPVI function
nPVI :: [Int] -> Double
nPVI xs = normalizer xs * sum ( pairwiseTerms xs )
     where normalizer xs = 100 / fromIntegral ( ( length xs ) -
1 )
```



Task 8 - Historic Code: The Dit Dah Code

Subtask 8a

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

```
>>> assoc 'a' symbols
('a',"- ---")
>>> assoc 'b' symbols
('b',"--- - - -")
>>> find 'c'
"--- - --- -"
>>> find 'd'
"--- - -"
>>>
```

Subtask 8c

```
>>> addletter 'a' 'b'

<interactive>:15:11: error:
    • Couldn't match expected type ‘[Char]’ with actual type ‘Char’
    • In the first argument of ‘addletter’, namely ‘'a'’
      In the expression: addletter 'a' 'b'
      In an equation for ‘it’: it = addletter 'a' 'b'

<interactive>:15:15: error:
    • Couldn't match expected type ‘[Char]’ with actual type ‘Char’
    • In the second argument of ‘addletter’, namely ‘'b'’
      In the expression: addletter 'a' 'b'
      In an equation for ‘it’: it = addletter 'a' 'b'
>>> addletter "a" "b"
"a   b"
>>> addword "joe" "tom"
"joe       tom"
>>> droplast3 "roses are red"
"roses are "
>>> droplast7 "car is green"
"car i"
>>>
```

Subtask 8d

```
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'a'
"_ ___"
>>> encodeletter 'b'
"___ _ _ _"
>>> encodeletter "yay"

<interactive>:23:14: error:
    • Couldn't match type ‘[Char]’ with ‘Char’
      Expected: Char
        Actual: String
    • In the first argument of ‘encodeletter’, namely ‘"yay"’
      In the expression: encodeletter "yay"
      In an equation for ‘it’: it = encodeletter "yay"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "hi"
"_ _ _ _   _ _"
>>> encodeword "hello"
"_ _ _ _   _   _ ___ _ _   _ ___ _ _   ___ ___ ___"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _       ___ _ __
_ _   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
>>> encodemessage "need time"
"___ _   _   _   ___ _ _       ___   _ _   ___ ___   _"
>>> encodemessage "translate the sentence"
"___   _ ___ _   _ ___   ___ _   _ _ _   _ ___ _ _   _ ___   ___   _       ___
_ _ _ _   _       _ _ _   _   ___ _   ___   _   ___ _   ___ _ ___ _   _"
>>>
```