Declan ONUNKWO 05/05/2023

CSC344 – Programming Languages

Assignment: **Haskell Programming Assignment – Various Computations**

LEARNING ABSTRACT

The project is aimed to further our knowledge on Haskell programming language by solving problems based on recursion, list comprehension, higher order functions and other fun tasks. With each task I was able to learn new Haskell functions and their usage, the syntax of the language, and so on. I also ran into errors which helped me gain more understanding of the language, what to do, and what not to do. This project helped me to develop a deeper appreciation for the power and elegance of Haskell, as well as to improve my problem-solving abilities and programming skills.

Task 1: Mindfully Mimicking the Demo

```
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"

<interactive>:15:20: error:
    • Couldn't match type ‘[Char]’ with ‘Char’
      Expected: Char
        Actual: String
    • In the first argument of ‘\ x -> x /= ' '’, namely ‘"Q"’
      In the expression: (\ x -> x /= ' ') "Q"
      In an equation for ‘it’: it = (\ x -> x /= ' ') "Q"
>>>( \x -> x /= ' ' ) 'Q'
True
>>>( \x -> x /= ' ' ) ' '
False
>>>filter ( \x -> x /= ' ' ) "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
>>>
```

Task 2: Numeric Function Definitions

The Code

declan — vim ha.hs — 80×25

```
squareArea l = l * l
circleArea r = pi * r * r
blueAreaOfCube l = 6 * ( (squareArea l) - (circleArea (0.25*l) ) )
paintedCube1 1 = 0
paintedCube1 n = 6 * ( squareArea (n - 2) )
paintedCube2 1 = 0
paintedCube2 n = 12 * (n - 2)
```

The Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :load ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>>"
>>>squareArea 10
100
>>>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]
>>>:q
Leaving GHCi.
```

Task 3: Puzzlers

The Code

```
 9 ------------------------------------------------------------------------
10
11 reverseWords userString = unwords(reverse(words userString))
12 averageWordLength userString = numChars/numWords
13                         where numChars = fromIntegral((length(filter (\x -> x /= ' ') userString)))
14                               numWords = fromIntegral(length(words userString))
```

The Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>>"
>>>:l ha
[1 of 1] Compiling Main             ( ha.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"
>>>reverseWords "is Haskell fun yet? Kinda"
"Kinda yet? fun Haskell is"
>>>reverseWords "Study shows that you pass your exams if you read your books"
"books your read you if exams your pass you that shows Study"
>>>:l ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>>averageWordLength "appa and baby yoda are the best"
3.5714285714285716
>>>averageWordLength "want me some coffee"
4.0
>>>averageWordLength "is Haskell fun yet? Kinda"
4.2
>>>averageWordLength "Study shows that you pass your exams if you read your books"
4.0
>>>:q
Leaving GHCi.
```

Task 4: Recursive List Processors

The Code

```haskell
-------------------------------------------------------------------------------------------------------------

list2set a =           ---------------------- Task 4a
  if (length a) == 0
    then []
    else if elem (head a) (list2set (tail a))
          then list2set (tail a)
          else (head a) : list2set (tail a)

isPalindrome [] = True ---------------------- Task 4b
isPalindrome [_] = True
isPalindrome (x:xs) =
   if (x == last xs)
   then isPalindrome (init xs)
   else False

collatz :: Integer -> [Integer] -------------- Task 4c
collatz n =
        if (n == 1)
        then [1]
        else if ((rem n 2) == 0)
             then n : collatz(div n 2)
             else n : collatz((3 * n) + 1)

-------------------------------------------------------------------------------------------------------------
```

The Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :l ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>>"
>>>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]
>>>:q
Leaving GHCi.
```

Task 5: List Comprehensions

The Code

```haskell

count a b = length (filter (\x -> x == a) b) ----------------- Task 5a

freqTable f = [(x,(count x f)) | x <- list2set f] ------------ Task 5b

-------------------------------------------------------------------------
```

The Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :l ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>>"
>>>count 'e' "need more coffee"
5
>>>count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
3
>>>count 's' "eat, sleep, study, repeat"
2
>>>count 6 [3,1,4,1,5,9,2,6,5,3,5,8]
1
>>>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)]
>>>freqTable "eat, sleep, study, repeat"
[('l',1),('s',2),('u',1),('d',1),('y',1),(',',3),(' ',3),('r',1),('p',2),('e',5),('a',2),('t',3)]
>>>freqTable [3,1,4,1,5,9,2,6,5,3,5,8]
[(4,1),(1,2),(9,1),(2,1),(6,1),(3,2),(5,3),(8,1)]
>>>:q
Leaving GHCi.
```

Task 6: Higher Order Functions

The Code

```haskell
-------------------------------------------------------------------------------------------------------------

                                        ------ Task 6 ------
tgl n = foldl (+) 0 [1..n]

triangleSequence n = map tgl [1..n]

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

lcsim mp fltr elmts = map (mp) (filter (fltr) elmts)

-------------------------------------------------------------------------------------------------------------
_
```

The Demo

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :l ha
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>>"
>>>:set prompt ">>> "
>>> 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]
>>>
```

Task 7: An Interesting Statistic - nPVI

Task 7a - The Test Data Code

~ — vim npvi.hs | ~ — ghc-9.2.7 -B/Users/d

```haskell
-------------------------------------------------------------------------------------------------
--                                          TASK 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]

```

Task 7b – The pairwiseValues function Code

```haskell

-------------------------------------------------------------------------------------------------
--                                          TASK 7B                                            --
-------------------------------------------------------------------------------------------------

pairwiseValues :: [Int] -> [(Int,Int)]

pairwiseValues var = zip var (tail var)

```

Task 7b – The pairwiseValues function Demo

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

```haskell
-- ------------------------------------------------------------------------------------------------
--                                          TASK 7C                                               --
-- ------------------------------------------------------------------------------------------------

pairwiseDifferences :: [Int] -> [Int]

pairwiseDifferences var = map (\(x,y) -> x - y) (pairwiseValues var)
```

7c – The pairwiseDifferences function Demo

```
>>> :l npvi
[1 of 1] Compiling Main             ( 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 function Code

```haskell
-- ------------------------------------------------------------------------------------------------
--                                          TASK 7D                                               --
-- ------------------------------------------------------------------------------------------------

pairwiseSums :: [Int] -> [Int]

pairwiseSums var = map (\(x,y) -> x + y) (pairwiseValues var)

```

Task 7d – The pairwiseSums function Demo

```
>>> :l npvi
[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

```haskell
-------------------------------------------------------------------------------------------------
--                                          TASK 7E                                            --
-------------------------------------------------------------------------------------------------

half :: Int -> Double
half number = ( fromIntegral number ) / 2

pairwiseHalves :: [Int] -> [Double]

pairwiseHalves var = map half var

```

Task 7e – The pairwiseHalves function Demo

```
>>> :l npvi
[1 of 1] Compiling Main             ( 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

```haskell
-------------------------------------------------------------------------------------------------
--                                          TASK 7F                                            --
-------------------------------------------------------------------------------------------------

pairwiseHalfSums :: [Int] -> [Double]

pairwiseHalfSums var = pairwiseHalves (pairwiseSums var)

```

Task 7f – The pairwiseHalfSums function Demo

```
>>> :l npvi
[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

```
66 ------------------------------------------------------------------------------------------------
67 --                                        TASK 7G                                             --
68 ------------------------------------------------------------------------------------------------
69
70 pairwiseTermPairs :: [Int] -> [(Int,Double)]
71
72 pairwiseTermPairs var = zip (pairwiseDifferences var) (pairwiseHalfSums var)
73
```

Task 7g – The pairwiseTermPairs function Demo

```
>>> :l npvi
[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

```
74 ------------------------------------------------------------------------------------------------
75 --                                        TASK 7H                                             --
76 ------------------------------------------------------------------------------------------------
77
78 term :: (Int,Double) -> Double
79 term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )
80
81 pairwiseTerms :: [Int] -> [Double]
82
83 pairwiseTerms val = map term (pairwiseTermPairs val)
84
```

Task 7h – The pairwiseTerms function Demo

```
>>> :l npvi
[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
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666]
>>> 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

```
85 ------------------------------------------------------------------------------------------
86 --                                      TASK 7I                                         --
87 ------------------------------------------------------------------------------------------
88
89 nPVI :: [Int] -> Double
90
91 nPVI xs = normalizer xs * sum ( pairwiseTerms xs )
92     where normalizer xs = 100 / fromIntegral ( ( length xs ) - 1 )
```

Task 7i – The nPVI function Demo

```
>>> :l npvi
[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

```
GHCi, version 9.2.7: https://www.haskell.org/ghc/  :? for help
ghci> :l ditdah
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> :set prompt ">>> "
>>> 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',"--- --- - -")]
>>>
```

Subtask 8b

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

Subtask 8c

```
>>> addletter (addletter (find 's') (find 'o')) (find 's') -- distress signal :)
"- - - --- --- --- - - -"
>>> addword (addletter (find 'h') (find 'i')) "there" -- hi there
"- - - - - - there"
>>> droplast3 "They love meat."
"They love me"
>>> droplast7 "Learning is fun"
"Learning"
>>>
```

Subtask 8d

```
>>> encodeletter 'm'
"--- ---"
>>> encodeletter 'v'
"- - - ---"
>>> encodeletter 'r'
"- --- -"
>>> encodeword "yay"
"--- - --- --- - --- --- - --- -"
>>> encodeword "sos"
"- - - --- --- --- - -"
>>> encodeword "lol"
"- --- - - --- --- --- - --- -"
>>> encodemessage "need more coffee"
"--- - - - --- - --- --- --- --- --- - --- - --- - --- - --- --- --- - - --- - - - -"
>>> encodemessage "need less coffee"
"--- - - - --- - - --- - - - - - - - - --- - --- - --- --- --- - - --- - - - -"
>>> encodemessage "finally finished"
"- - --- - - - --- - - --- - --- - - - --- - - --- - --- - - - --- - - - --- - - - - - - - - - - -"
>>>
```