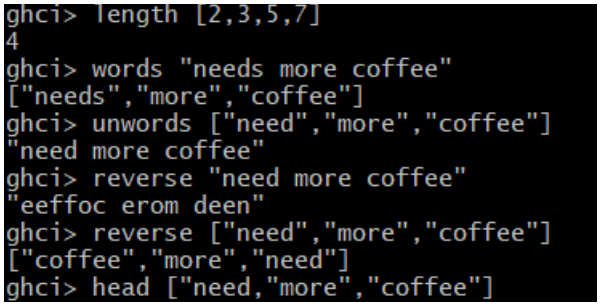
Assignment #7

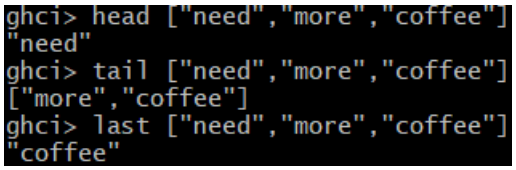
Trevor Strickland, CSC344

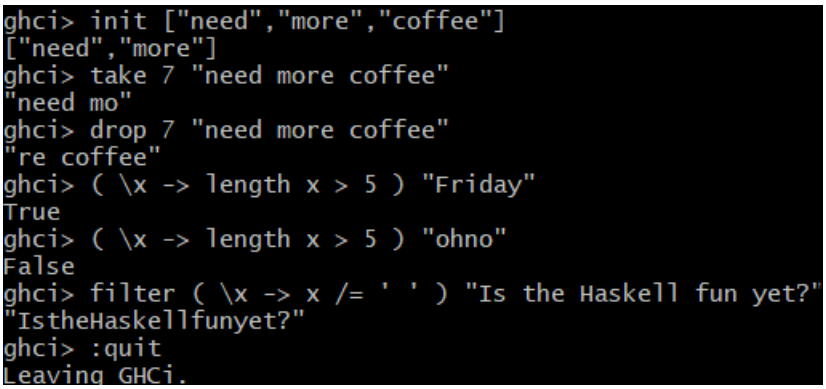
November 28, 2022

Learning Abstract: Like the previous assignment, this assignment serves as an introduction to a new language, this time Haskell. This assignment runs down the basics of Haskell syntax and list processing, as well as how to reconstruct and solve the nPVI formula in code and decoding morse.

Task 1: Mindfully Mimicking the Demo





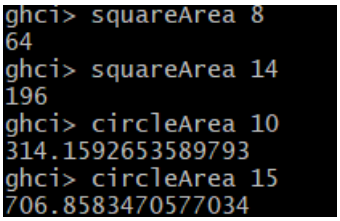


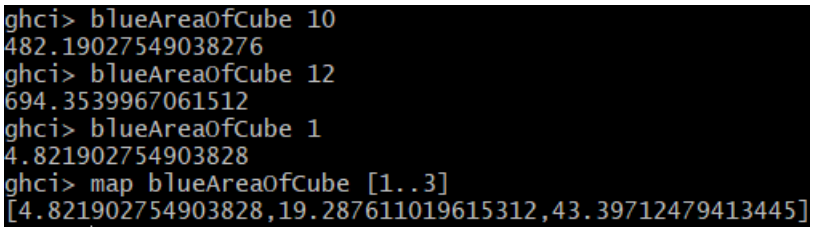
Task 2: Numeric Function Definitions

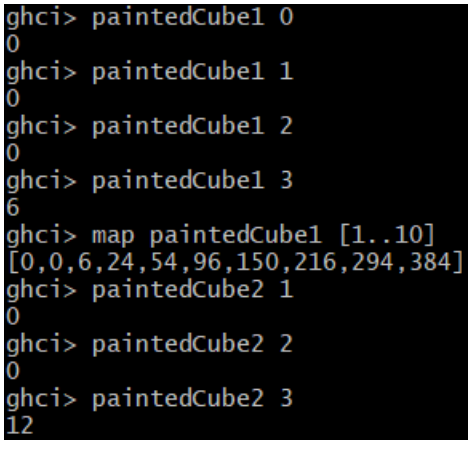
Code:

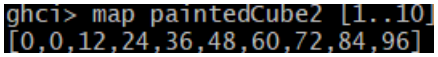
```
--Task 2

squareArea x = x * x

circleArea x = x * x * pi

blueAreaOfCube x = 6 * (squareArea x - circleArea (x / 4))

paintedCube1 n =
   if n > 2 then 6 * squareArea (n - 2)
   else 0

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

Demo:









Task 3: Puzzlers:

Code:

```
--Task 3

reverseWords l = unwords (reverse (words l))

averageWordLength l = a / b
   where a = fromIntegral (foldr (+) 0 (map length (words l)))
         b = fromIntegral (length (words l))
```

Demo:

```
ghci> reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
ghci> reverseWords "coffee some me want"
"want me some coffee"
ghci> reverseWords "try this one for size mister"
"mister size for one this try"
```

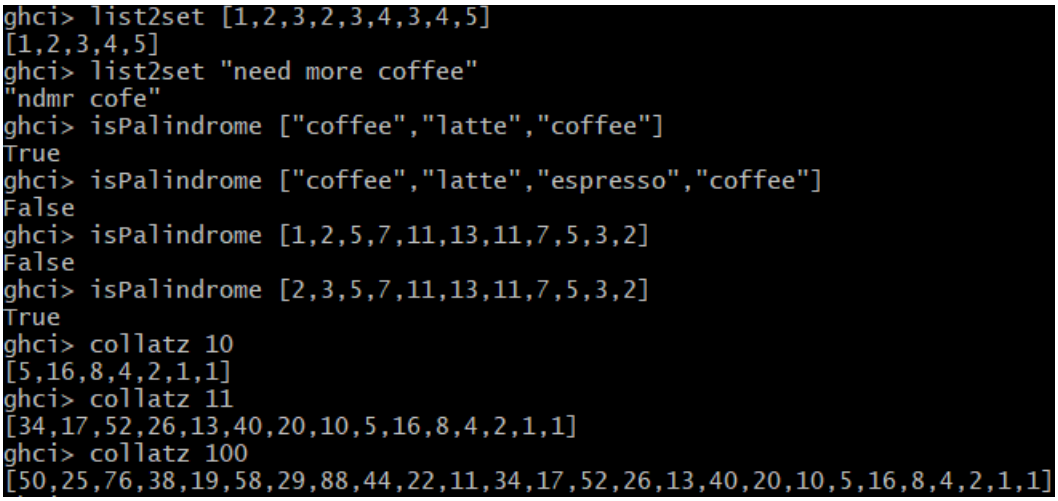
```
ghci> averageWordLength "appa and baby yoda are the best"
3.5714285714285716
ghci> averageWordLength "want me some coffee"
4.0
ghci> averageWordLength "try this one for size mister"
3.8333333333333335
```

Task 4: Recursive List Processors

Code:

```
--Task 4

list2set [] = []
list2set [x] = [x]

list2set l =
   if (elem (head l) (tail l)) then list2set (tail l)
   else (head l) : (list2set (tail l))

isPalindrome [] = True
isPalindrome [_] = True

isPalindrome l =
   if (head l) == (last l) then isPalindrome(init(tail l))
   else False

collatz x =
   if x == 1 then [1]
   else
      if odd x then [(3 * x + 1)] ++ collatz (3 * x + 1)
      else [div x 2] ++ collatz (div x 2)
```

Demo:

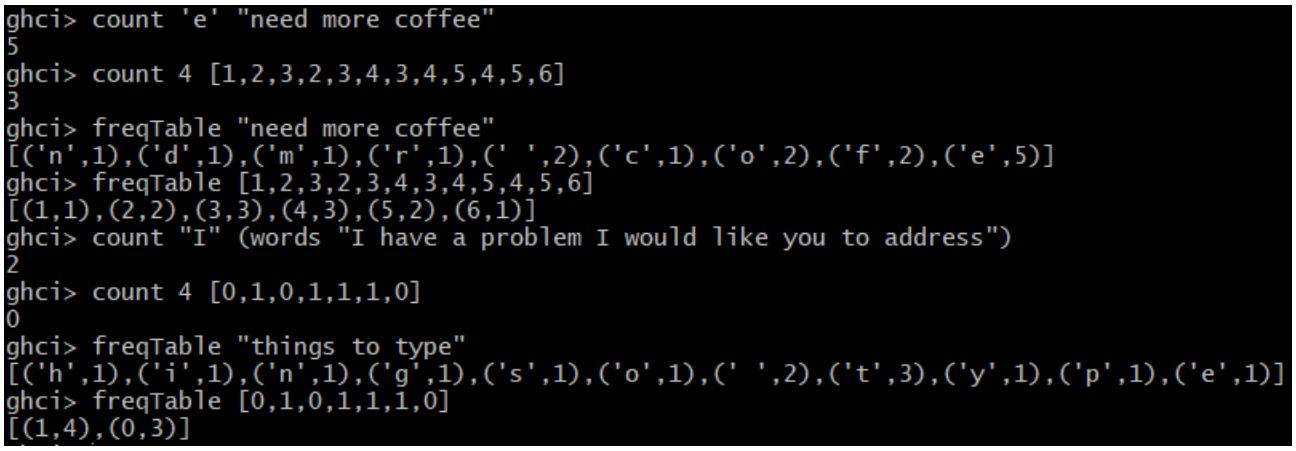


Task 5: List Comprehensions

Code:

```
--Task 5

count x l = length [a | a ← l, a == x]
freqTable l = [(x, count x l) | x ← list2set l]
```

Demo:



Task 6: Higher Order Functions

Code:

```
--Task 6

tgl x = foldr (+) 0 [1..x]
triangleSequence x = map tgl [1..x]

vowelCount x = length (filter (\x → x `elem` "aeiou") x)

lcsim func pred l = map func (filter pred l)
```

Demo:

```
ghci> tgl 5
15
ghci> tgl 55
1540
ghci> tgl 10
55
ghci> tgl 100
5050
ghci> triangleSequence 10
[1,3,6,10,15,21,28,36,45,55]
ghci> triangleSequence 20
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210]
ghci> triangleSequence 30
[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
,435,465]
ghci> triangleSequence 40
[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
,435,465,496,528,561,595,630,666,703,741,780,820]
```

```
ghci> vowelCount "cat"
1
ghci> vowelCount "mouse"
3
ghci> vowelCount "scrumptious"
4
ghci> vowelCount "scromblo bomblo"
4
ghci> lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
ghci> animals = ["elephant", "lion", "tiger", "orangatan", "jaguar"]
ghci> lcsim length (\x -> elem (head x) "aeiou") animals
[8,9]
ghci> lcsim tgl even [1..15]
[3,10,21,36,55,78,105]
ghci> lcsim tgl even [1..100]
[3,10,21,36,55,78,105,136,171,210,253,300,351,406,465,528,595,666,741,820,903,990,1081,1176,1275
,1378,1485,1596,1711,1830,1953,2080,2211,2346,2485,2628,2775,2926,3081,3240,3403,3570,3741,3916,
4095,4278,4465,4656,4851,5050]
```

Task 7: nPVI

Code & Demo:

```
-- Test data (7a)

a :: [Int]
a = [2,5,1,3]

b :: [Int]
b = [1,3,6,2,5]

c :: [Int]
c = [4,4,2,1,1,2,2,4,4,4,8]

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

pairwiseValues :: [Int] → [(Int, Int)]
pairwiseValues x = zip x (tail x)
```

```
ghci> pairwiseValues a
[(2,5),(5,1),(1,3)]
ghci> pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
ghci> pairwiseValues c
[(4,4),(4,2),(2,1),(1,1),(1,2),(2,2),(2,4),(4,4),(4,4),(4,8)]
ghci> pairwiseValues u
[(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2)]
ghci> pairwiseValues x
[(1,9),(9,2),(2,8),(8,3),(3,7),(7,2),(2,8),(8,1),(1,9)]
```

```
--7c

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

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

```
--7d

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

```
ghci> pairwiseSums a
[7,6,4]
ghci> pairwiseSums b
[4,9,8,7]
ghci> pairwiseSums c
[8,6,3,2,3,4,6,8,8,12]
ghci> pairwiseSums u
[4,4,4,4,4,4,4,4,4,4]
ghci> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
```

```
--7e

half :: Int → Double
half x = (fromIntegral x) / 2

pairwiseHalves :: [Int] → [Double]
pairwiseHalves x = map half x
```

```
ghci> pairwiseHalves [1..10]
[0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
ghci> pairwiseHalves u
[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
ghci> 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 x = map half (pairwiseSums x)
```

```
ghci> pairwiseHalfSums a
[3.5,3.0,2.0]
ghci> pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
ghci> pairwiseHalfSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,4.0,6.0]
ghci> pairwiseHalfSums u
[2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0]
ghci> 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 x = zip (pairwiseDifferences x) (pairwiseHalfSums x)
```

```
ghci> pairwiseTermPairs a
[(-3,3.5),(4,3.0),(-2,2.0)]
ghci> pairwiseTermPairs b
[(-2,2.0),(-3,4.5),(4,4.0),(-3,3.5)]
ghci> 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),(0,4.0),(-4,6.0)]
ghci> 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)]
ghci> 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

```
term :: (Int, Double) → Double
term x = abs (fromIntegral (fst x) / (snd x))

pairwiseTerms :: [Int] → [Double]
pairwiseTerms x = map term (pairwiseTermPairs x)
```

```
ghci> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
ghci> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
ghci> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.0
,0.6666666666666666]
ghci> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
ghci> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555555555555556,1.6
]
```

--7i

```
nPVI :: [Int] → Double
nPVI xs = normalizer xs * sum (pairwiseTerms xs)
    where normalizer xs = 100 / fromIntegral ((length xs) - 1)
```

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

Task 8: The Dit Dah Code

Demo:

```
ghci> dit
"_"
ghci> dah
"___"
ghci> "a" +++ "b"
"a b"
ghci> m
('m',"--- ---")
ghci> g
('g',"--- --- -")
ghci> h
('h',"- - - -")
ghci> 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',"--- --- - -")]
```

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

```
ghci> addletter "a" "b"
"a   b"
ghci> addword "apple" "banana"
"apple       banana"
ghci> droplast3 "the last 3"
"the las"
ghci> droplast7 "the last 7"
"the"
```

```
ghci> encodeletter 'm'
"--- ---"
ghci> encodeletter 'f'
"- - --- -"
ghci> encodeletter 'c'
"--- - --- -"
ghci> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
ghci> encodeword "nay"
"--- -   - ---   --- - --- ---"
ghci> encodeword "may"
"--- ---   - ---   --- - --- ---"
ghci> encodemessage "need more coffee"
"--- -   -   -   --- - -       --- ---   --- --- ---   - --- -   -       --- - --- -   --- --- -
--   - - --- -   - - --- -   -   -"
ghci> encodemessage "dont we all"
"--- - -   --- --- ---   --- -   ---       - --- ---   -       - ---   - --- - -   - --- - -"
ghci> encodemessage "yay assignment done"
"--- - --- ---   - ---   --- - --- ---       - ---   - - -   - - -   - -   --- --- -   --- -   -
-- ---   -   --- -   ---       --- - -   --- --- ---   --- -   -"
```