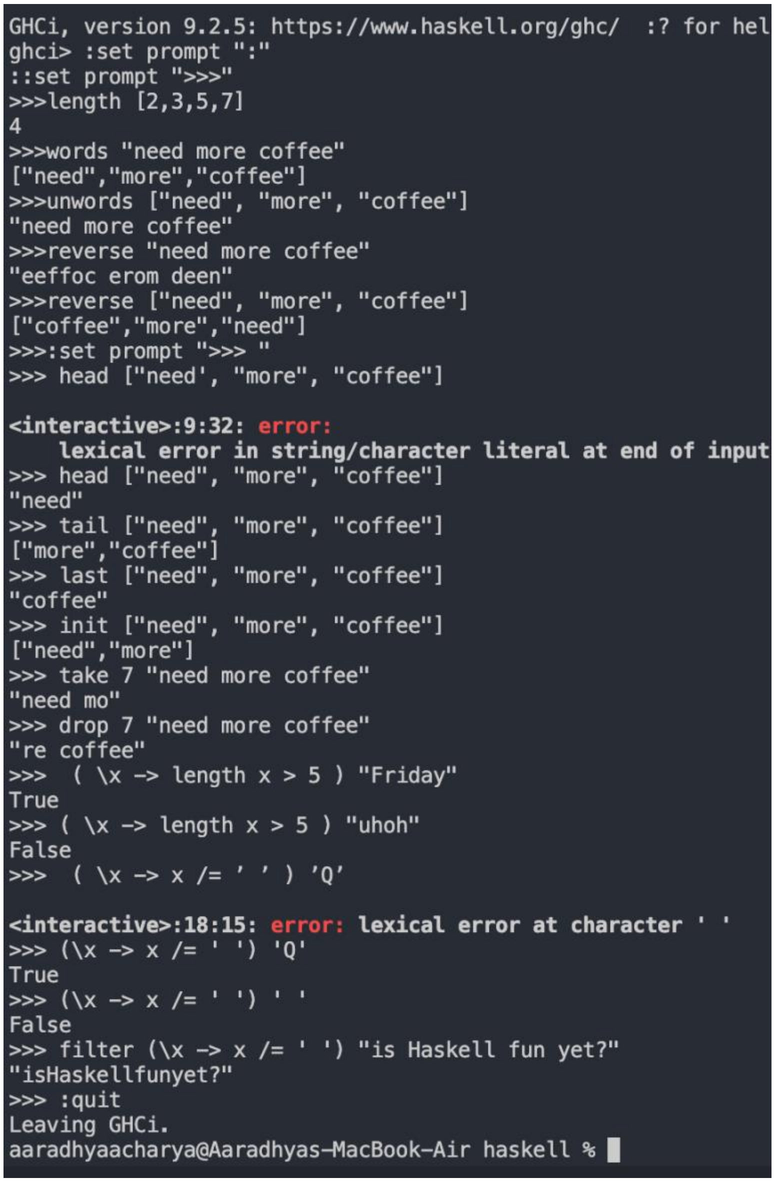
# Haskell Programming Assignment: Various Computations

## Learning Abstract

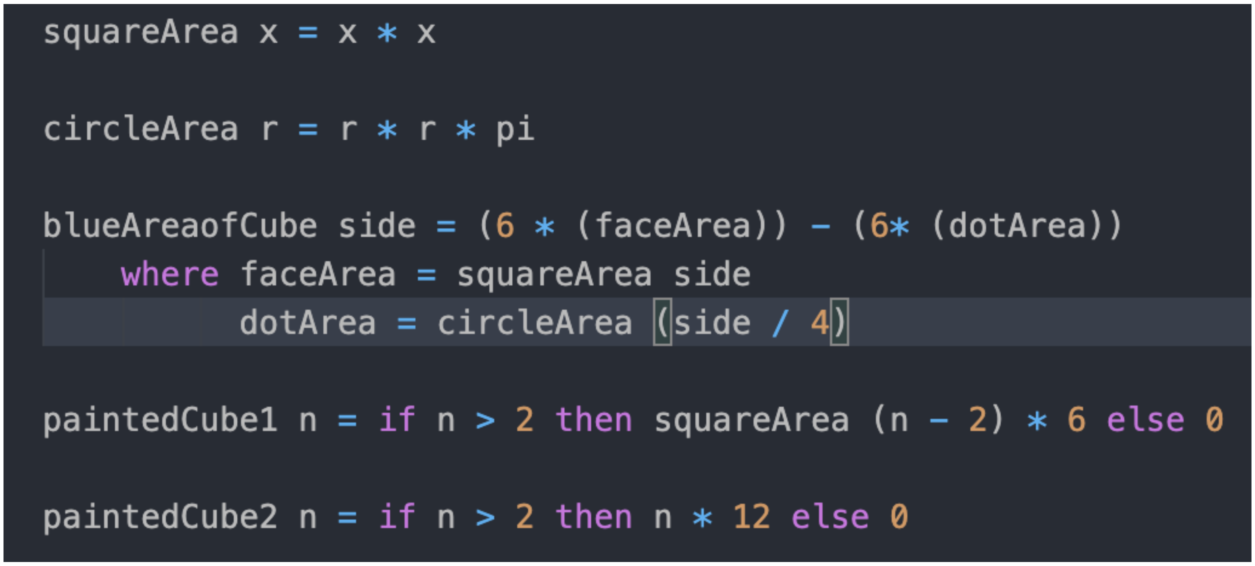
This assignment is the only assignment on Haskell and it features programming exercises that focus on functions, recursive list processing, list comprehensions, and higher order functions in Haskell.

## Task 1: Mindfully mimicking the demo

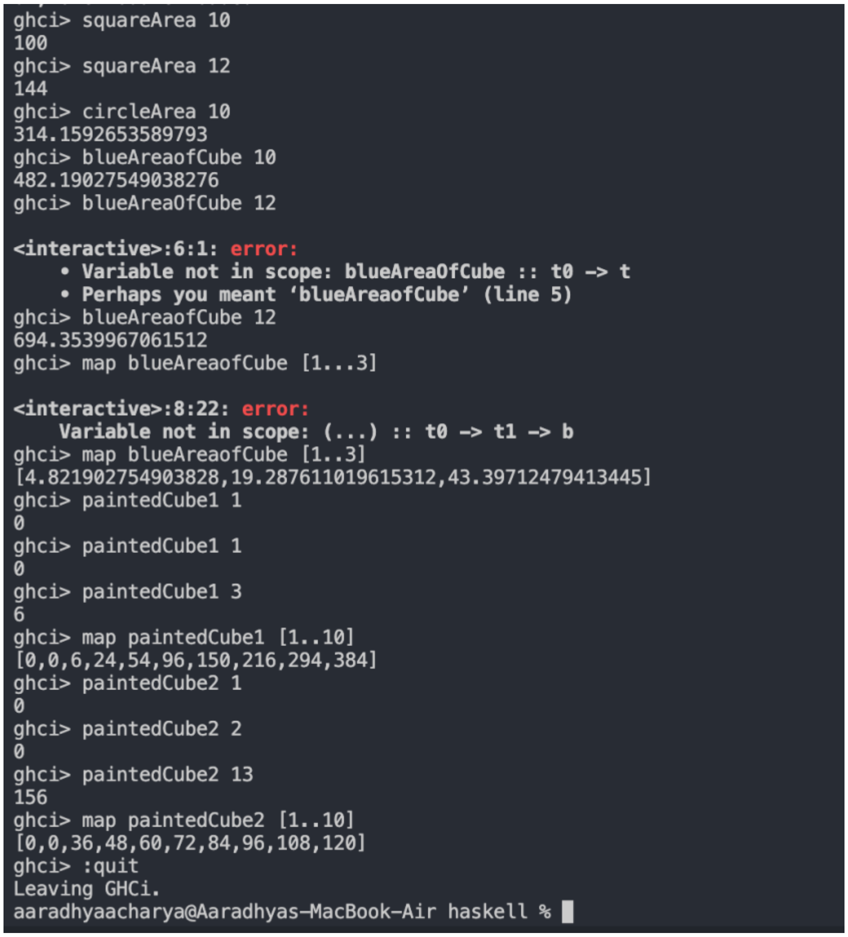


```
GHCi, version 9.2.5: https://www.haskell.org/ghc/  :? for hel
ghci> :set prompt ":"
::set prompt ">>>"
>>>length [2,3,5,7]
4
>>>words "need more coffee"
["need","more","coffee"]
>>>unwords ["need", "more", "coffee"]
"need more coffee"
>>>reverse "need more coffee"
"eeffoc erom deen"
>>>reverse ["need", "more", "coffee"]
["coffee","more","need"]
>>>:set prompt ">>> "
>>> head ["need', "more", "coffee"]

<interactive>:9:32: error:
    lexical error in string/character literal at end of input
>>> 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>:18:15: error: lexical error at character ' '
>>> (\x -> x /= ' ') 'Q'
True
>>> (\x -> x /= ' ') ' '
False
>>> filter (\x -> x /= ' ') "is Haskell fun yet?"
"isHaskellfunyet?"
>>> :quit
Leaving GHCi.
aaradhyaacharya@Aaradhyas-MacBook-Air haskell %
```

## Task 2: Numeric Function Definitions

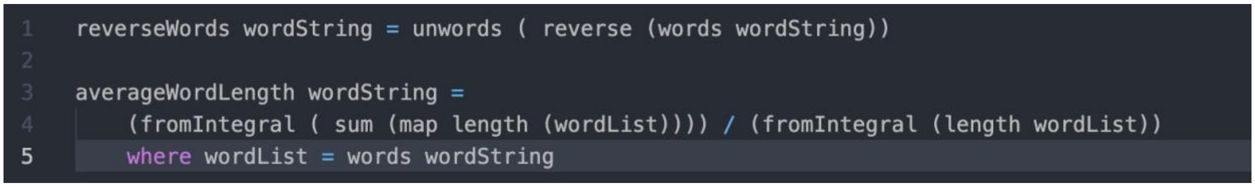


```haskell
squareArea x = x * x

circleArea r = r * r * pi

blueAreaofCube side = (6 * (faceArea)) - (6* (dotArea))
    where faceArea = squareArea side
          dotArea = circleArea (side / 4)

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

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

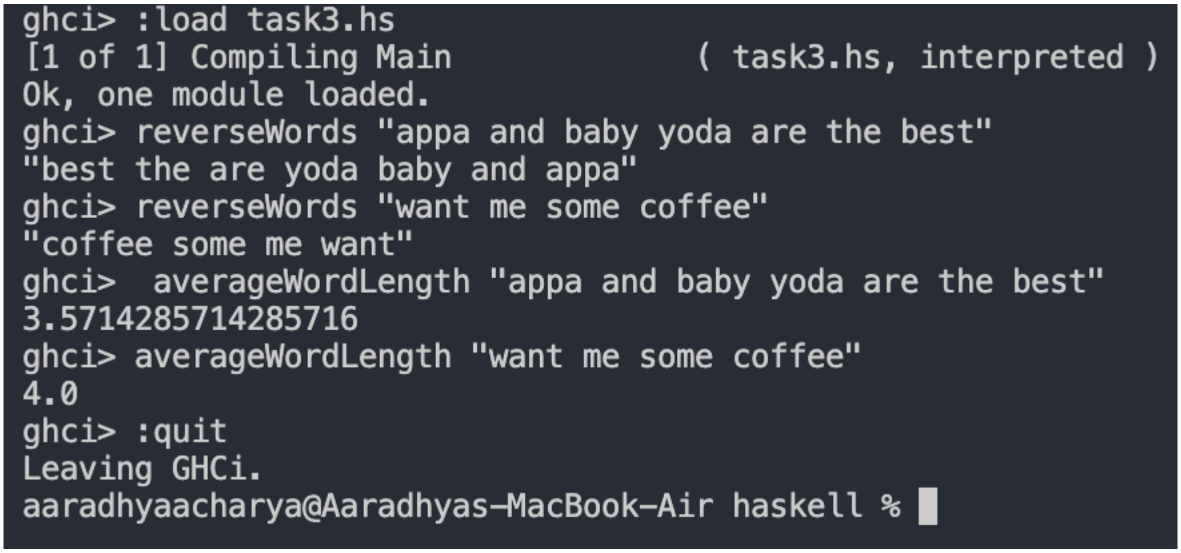


```
ghci> squareArea 10
100
ghci> squareArea 12
144
ghci> circleArea 10
314.1592653589793
ghci> blueAreaofCube 10
482.19027549038276
ghci> blueAreaOfCube 12

<interactive>:6:1: error:
    • Variable not in scope: blueAreaOfCube :: t0 -> t
    • Perhaps you meant ‘blueAreaofCube’ (line 5)
ghci> blueAreaofCube 12
694.3539967061512
ghci> map blueAreaofCube [1...3]

<interactive>:8:22: error:
    Variable not in scope: (...) :: t0 -> t1 -> b
ghci> map blueAreaofCube [1..3]
[4.821902754903828,19.287611019615312,43.39712479413445]
ghci> paintedCube1 1
0
ghci> paintedCube1 1
0
ghci> paintedCube1 3
6
ghci> map paintedCube1 [1..10]
[0,0,6,24,54,96,150,216,294,384]
ghci> paintedCube2 1
0
ghci> paintedCube2 2
0
ghci> paintedCube2 13
156
ghci> map paintedCube2 [1..10]
[0,0,36,48,60,72,84,96,108,120]
ghci> :quit
Leaving GHCi.
aaradhyaacharya@Aaradhyas-MacBook-Air haskell % 
```

## Task 3: Puzzlers

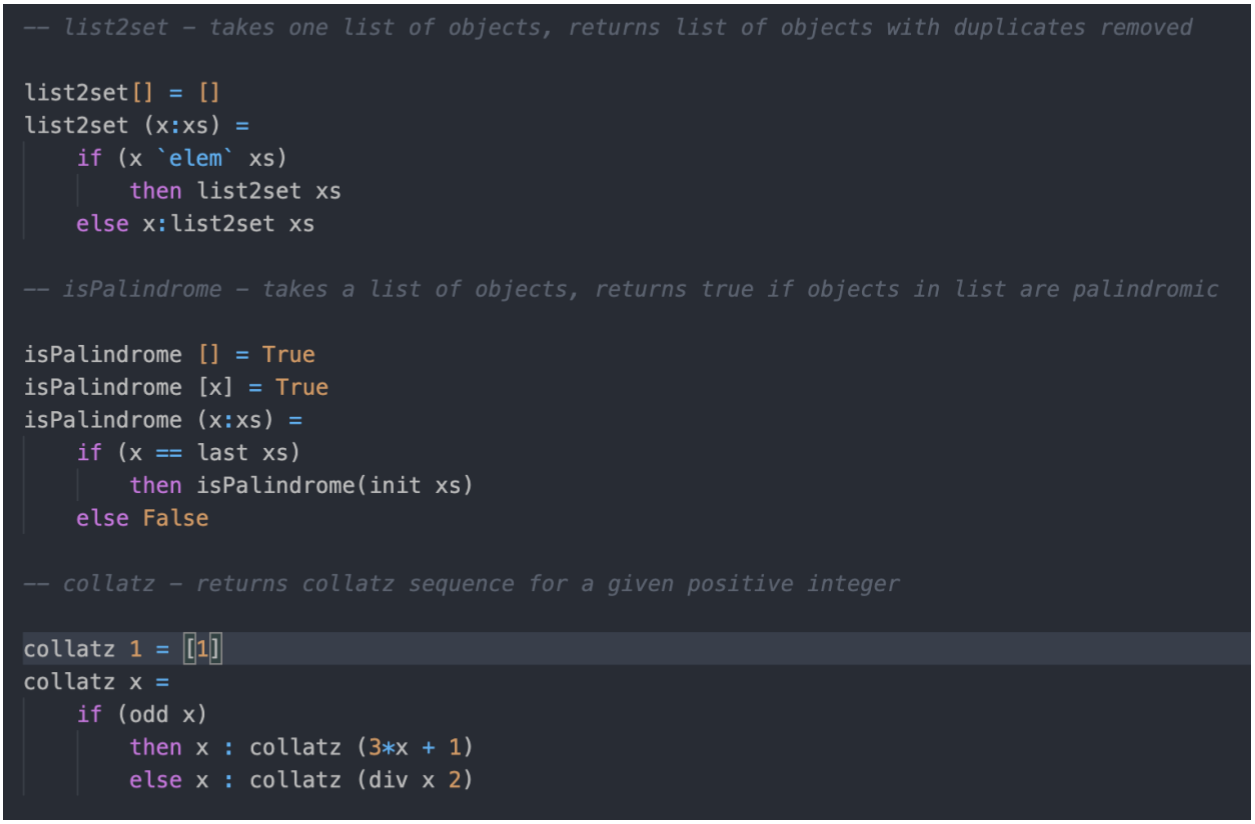


```haskell
reverseWords wordString = unwords ( reverse (words wordString))

averageWordLength wordString =
    (fromIntegral ( sum (map length (wordList)))) / (fromIntegral (length wordList))
    where wordList = words wordString
```

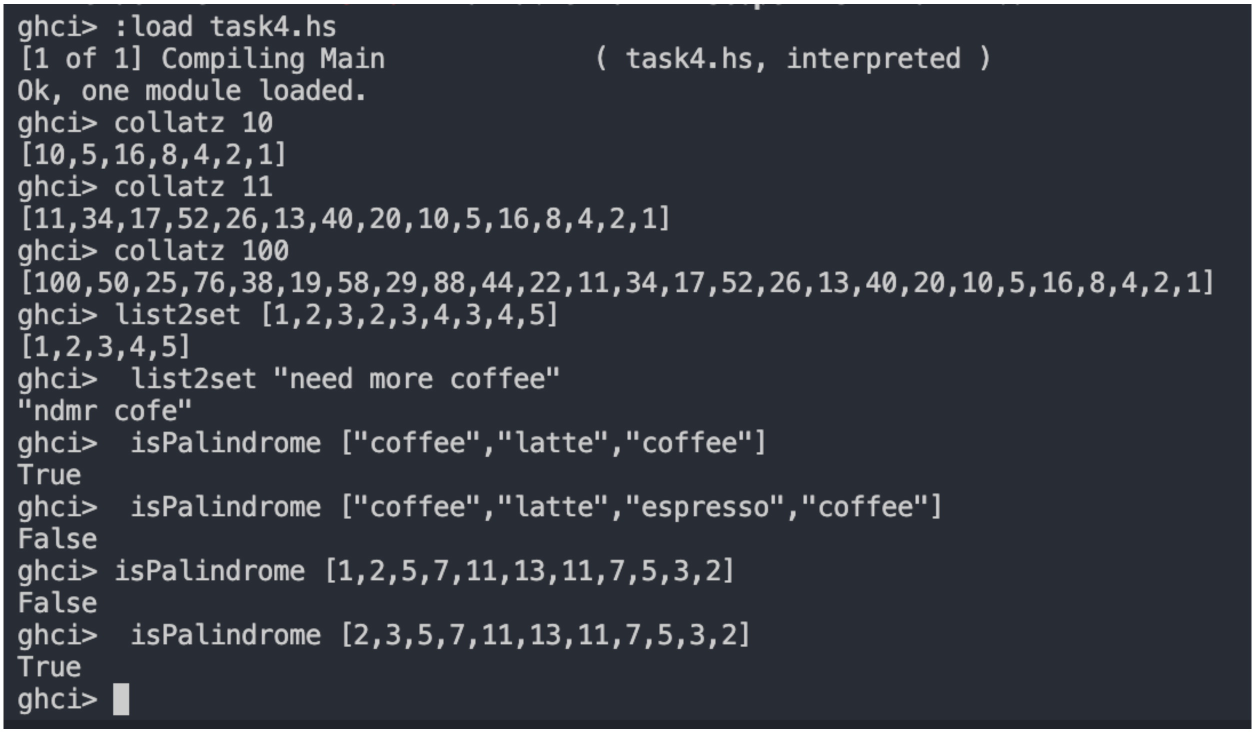


```
ghci> :load task3.hs
[1 of 1] Compiling Main             ( task3.hs, interpreted )
Ok, one module loaded.
ghci> reverseWords "appa and baby yoda are the best"
"best the are yoda baby and appa"
ghci> reverseWords "want me some coffee"
"coffee some me want"
ghci>  averageWordLength "appa and baby yoda are the best"
3.5714285714285716
ghci> averageWordLength "want me some coffee"
4.0
ghci> :quit
Leaving GHCi.
aaradhyaacharya@Aaradhyas-MacBook-Air haskell %
```

## Task 4: Recursive List Processors

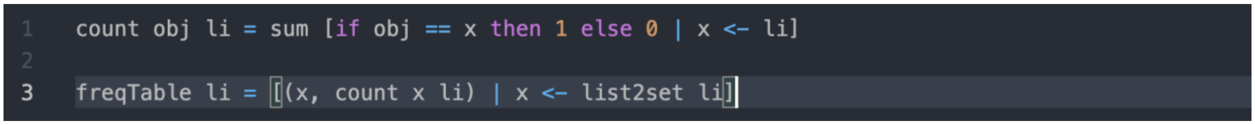


```haskell
-- list2set - takes one list of objects, returns list of objects with duplicates removed

list2set[] = []
list2set (x:xs) =
    if (x `elem` xs)
        then list2set xs
    else x:list2set xs

-- isPalindrome - takes a list of objects, returns true if objects in list are palindromic

isPalindrome [] = True
isPalindrome [x] = True
isPalindrome (x:xs) =
    if (x == last xs)
        then isPalindrome(init xs)
    else False

-- collatz - returns collatz sequence for a given positive integer

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

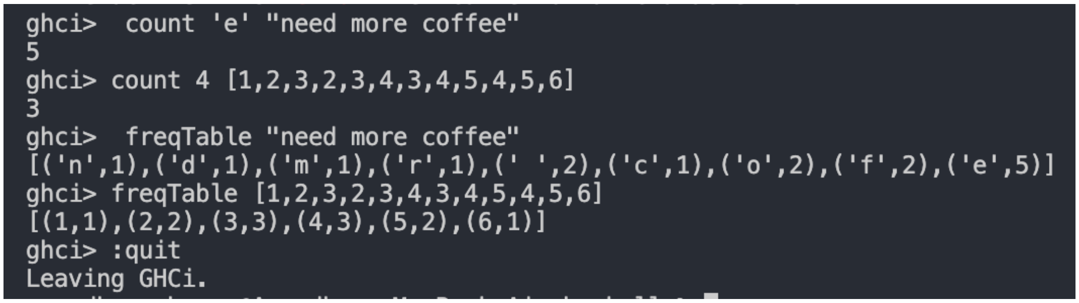


```
ghci> :load task4.hs
[1 of 1] Compiling Main             ( task4.hs, interpreted )
Ok, one module loaded.
ghci> collatz 10
[10,5,16,8,4,2,1]
ghci> collatz 11
[11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
ghci> collatz 100
[100,50,25,76,38,19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
ghci> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
ghci>  list2set "need more coffee"
"ndmr cofe"
ghci>  isPalindrome ["coffee","latte","coffee"]
True
ghci>  isPalindrome ["coffee","latte","espresso","coffee"]
False
ghci> isPalindrome [1,2,5,7,11,13,11,7,5,3,2]
False
ghci>  isPalindrome [2,3,5,7,11,13,11,7,5,3,2]
True
ghci> █
```

## Task 5: List Comprehensions

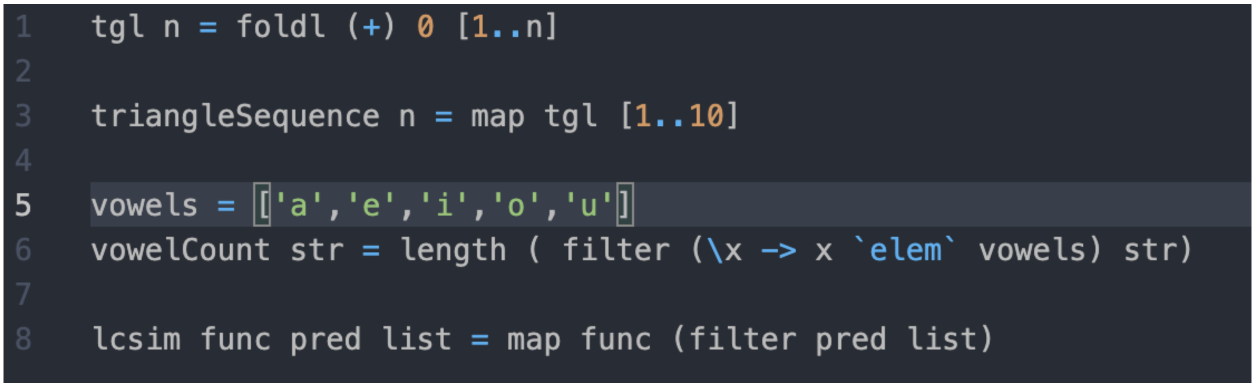


```haskell
count obj li = sum [if obj == x then 1 else 0 | x <- li]

freqTable li = [(x, count x li) | x <- list2set li]
```

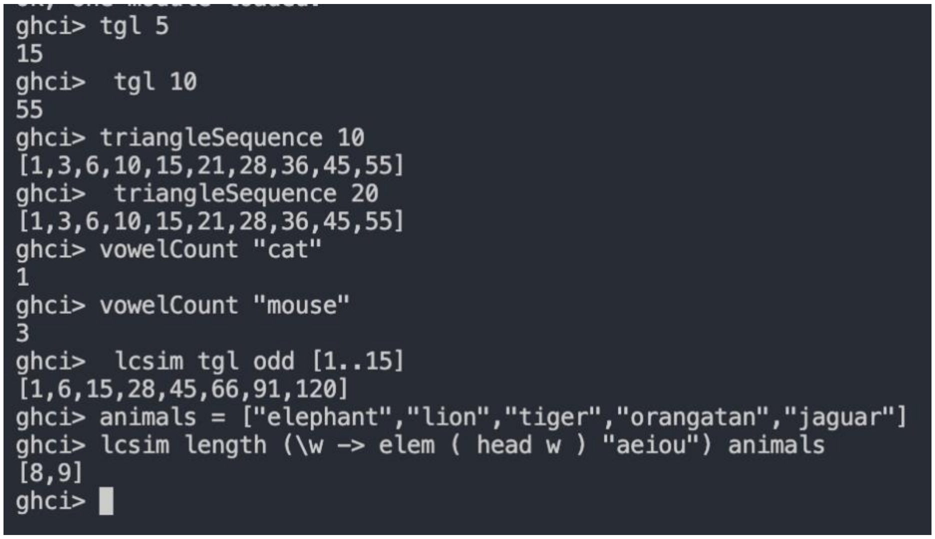


```
ghci>  count 'e' "need more coffee"
5
ghci> count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
3
ghci>  freqTable "need more coffee"
[('n',1),('d',1),('m',1),('r',1),(' ',2),('c',1),('o',2),('f',2),('e',5)]
ghci> freqTable [1,2,3,2,3,4,3,4,5,4,5,6]
[(1,1),(2,2),(3,3),(4,3),(5,2),(6,1)]
ghci> :quit
Leaving GHCi.
```

## Task 6:  Higher Order Functions

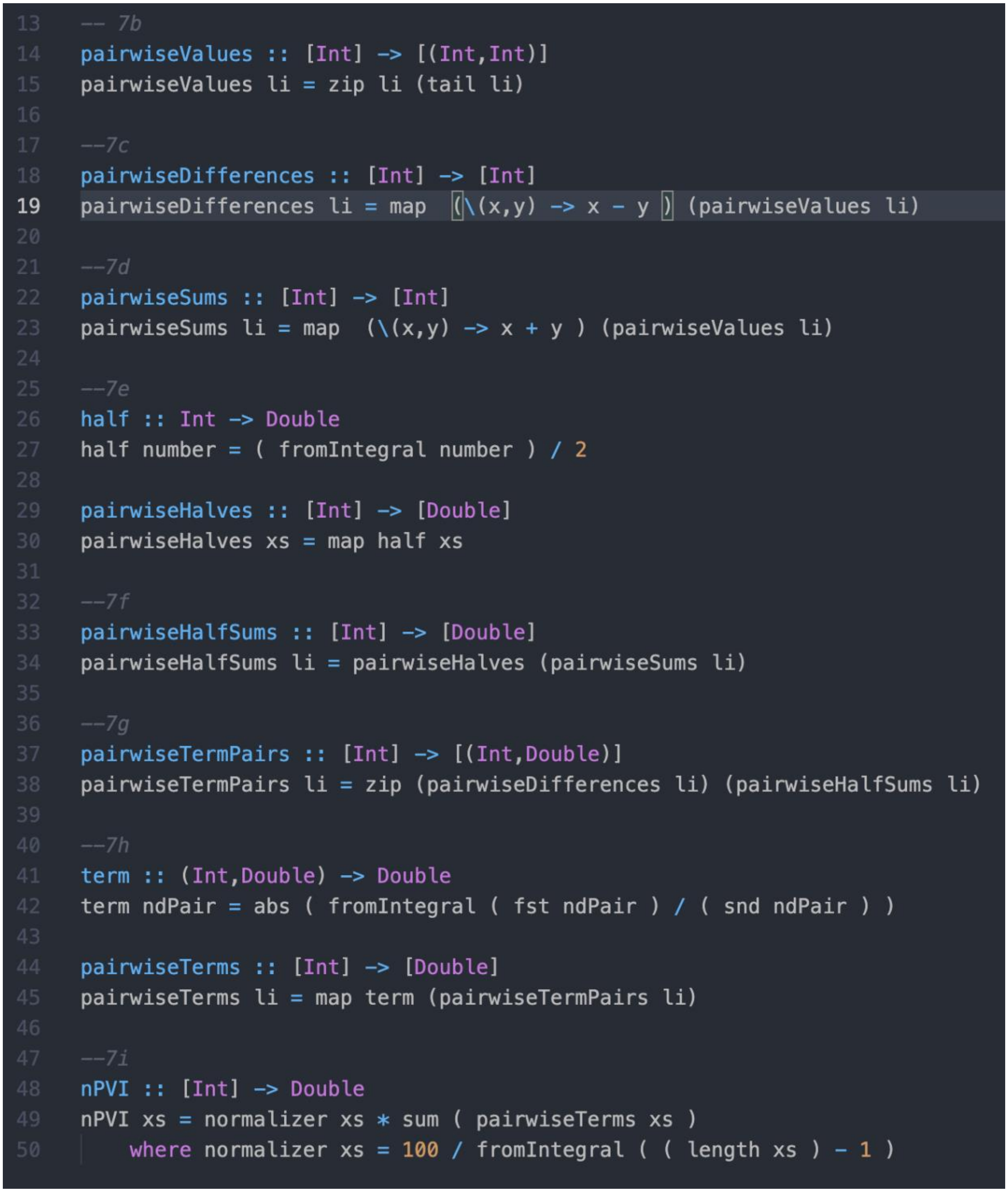


```haskell
tgl n = foldl (+) 0 [1..n]

triangleSequence n = map tgl [1..10]

vowels = ['a','e','i','o','u']
vowelCount str = length ( filter (\x -> x `elem` vowels) str)

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



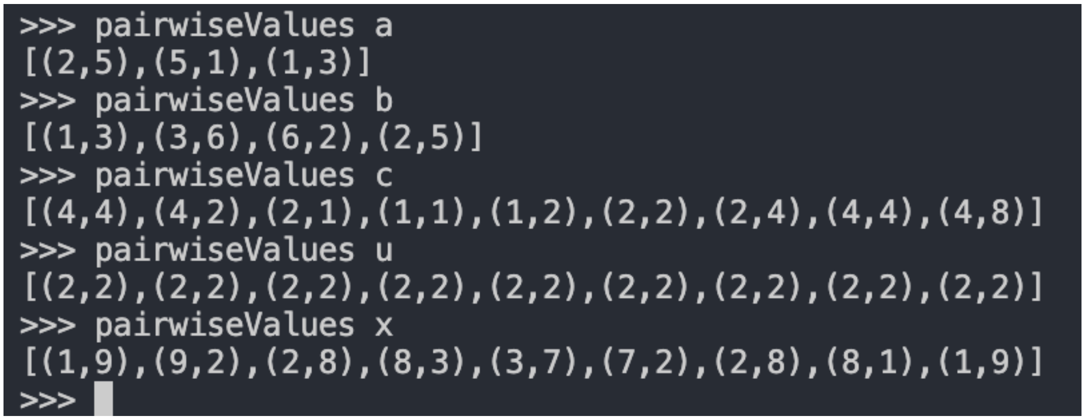
```
ghci> tgl 5
15
ghci>  tgl 10
55
ghci> triangleSequence 10
[1,3,6,10,15,21,28,36,45,55]
ghci>  triangleSequence 20
[1,3,6,10,15,21,28,36,45,55]
ghci> vowelCount "cat"
1
ghci> vowelCount "mouse"
3
ghci>  lcsim tgl odd [1..15]
[1,6,15,28,45,66,91,120]
ghci> animals = ["elephant","lion","tiger","orangatan","jaguar"]
ghci> lcsim length (\w -> elem ( head w ) "aeiou") animals
[8,9]
ghci>
```

## Task 7: An Interesting Statistic, nPVI



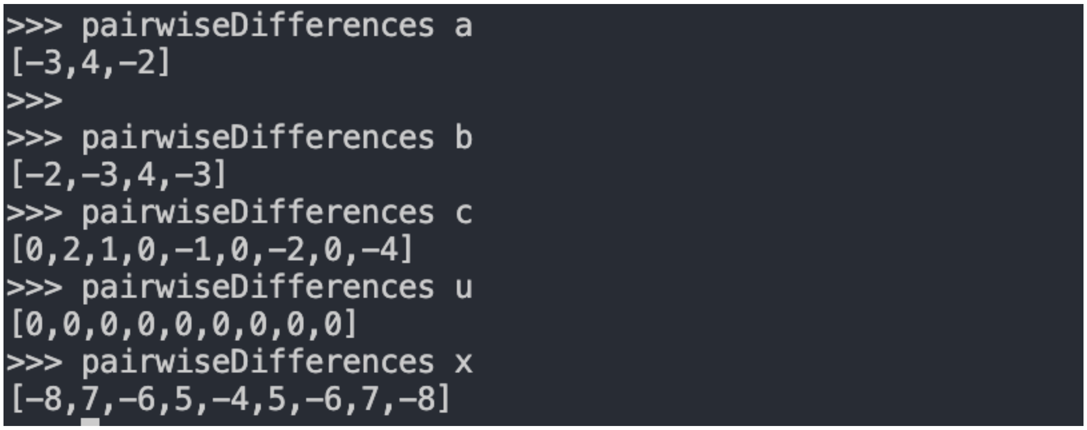
```haskell
-- 7b
pairwiseValues :: [Int] -> [(Int,Int)]
pairwiseValues li = zip li (tail li)

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

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

--7e
half :: Int -> Double
half number = ( fromIntegral number ) / 2

pairwiseHalves :: [Int] -> [Double]
pairwiseHalves xs = map half xs

--7f
pairwiseHalfSums :: [Int] -> [Double]
pairwiseHalfSums li = pairwiseHalves (pairwiseSums li)

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

--7h
term :: (Int,Double) -> Double
term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )

pairwiseTerms :: [Int] -> [Double]
pairwiseTerms li = map term (pairwiseTermPairs li)

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

7b



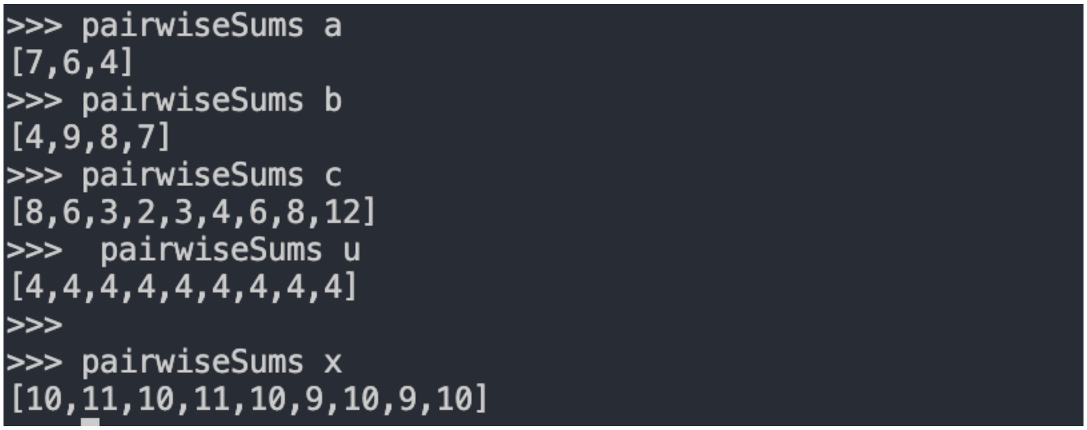
```
>>> pairwiseValues a
[(2,5),(5,1),(1,3)]
>>> pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
>>> pairwiseValues c
[(4,4),(4,2),(2,1),(1,1),(1,2),(2,2),(2,4),(4,4),(4,8)]
>>> pairwiseValues u
[(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2)]
>>> pairwiseValues x
[(1,9),(9,2),(2,8),(8,3),(3,7),(7,2),(2,8),(8,1),(1,9)]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

7c



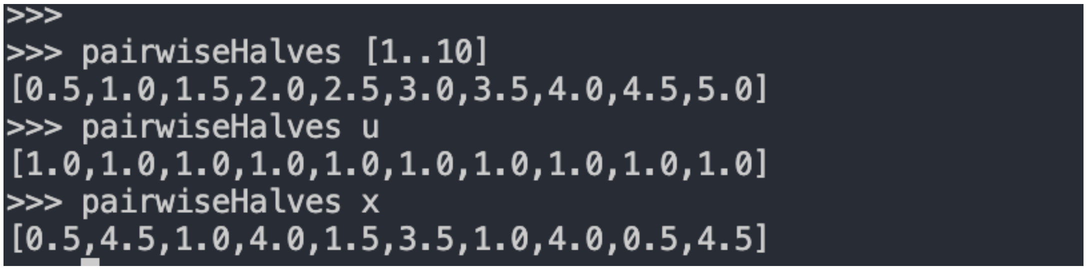
```
>>> pairwiseDifferences a
[-3,4,-2]
>>>
>>> pairwiseDifferences b
[-2,-3,4,-3]
>>> pairwiseDifferences c
[0,2,1,0,-1,0,-2,0,-4]
>>> pairwiseDifferences u
[0,0,0,0,0,0,0,0,0]
>>> pairwiseDifferences x
[-8,7,-6,5,-4,5,-6,7,-8]
```

7d



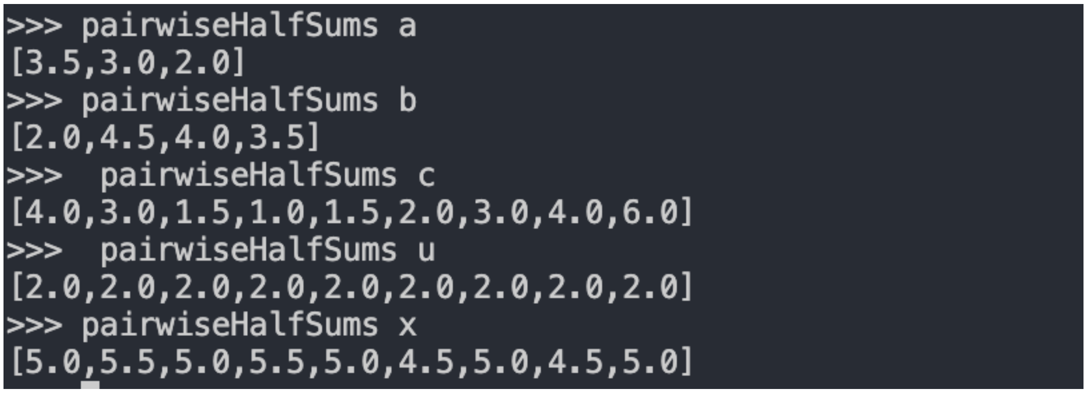
```
>>> pairwiseSums a
[7,6,4]
>>> pairwiseSums b
[4,9,8,7]
>>> pairwiseSums c
[8,6,3,2,3,4,6,8,12]
>>>  pairwiseSums u
[4,4,4,4,4,4,4,4,4]
>>>
>>> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
```

7e



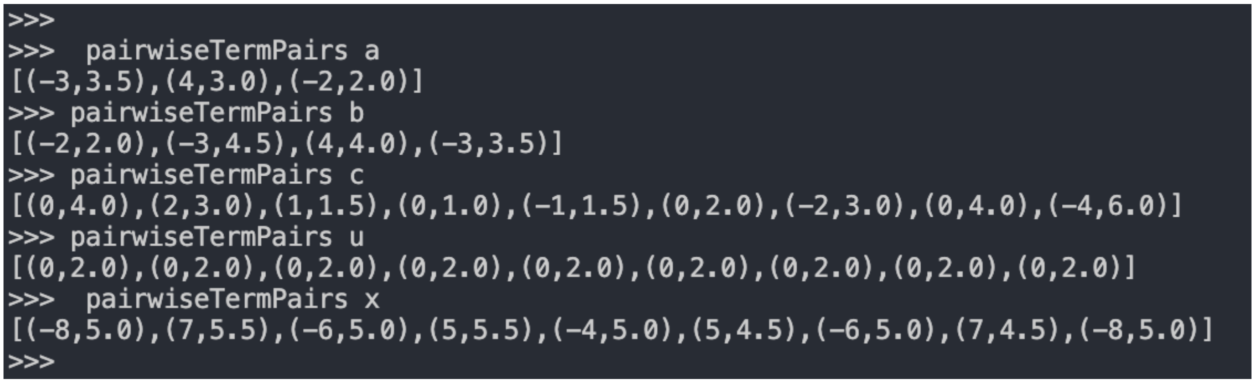
```
>>>
>>> pairwiseHalves [1..10]
[0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
>>> pairwiseHalves u
[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
>>> pairwiseHalves x
[0.5,4.5,1.0,4.0,1.5,3.5,1.0,4.0,0.5,4.5]
```

7f



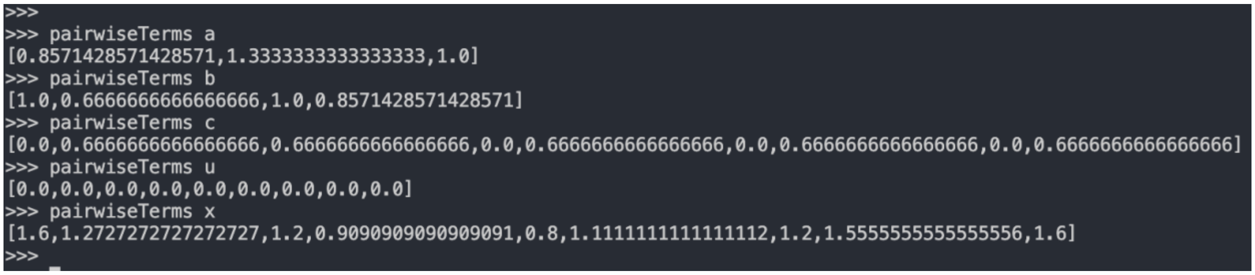
```
>>> pairwiseHalfSums a
[3.5,3.0,2.0]
>>> pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
>>>  pairwiseHalfSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,6.0]
>>>  pairwiseHalfSums u
[2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0]
>>> pairwiseHalfSums x
[5.0,5.5,5.0,5.5,5.0,4.5,5.0,4.5,5.0]
```

7g



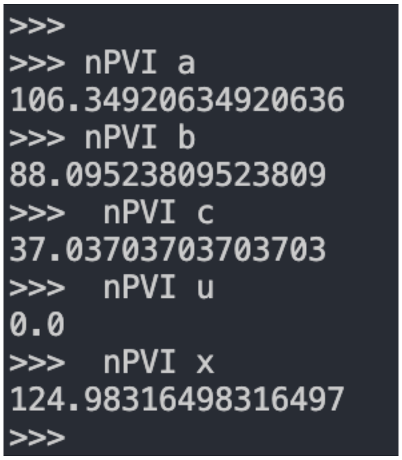
```
>>>
>>>  pairwiseTermPairs a
[(-3,3.5),(4,3.0),(-2,2.0)]
>>> pairwiseTermPairs b
[(-2,2.0),(-3,4.5),(4,4.0),(-3,3.5)]
>>> pairwiseTermPairs c
[(0,4.0),(2,3.0),(1,1.5),(0,1.0),(-1,1.5),(0,2.0),(-2,3.0),(0,4.0),(-4,6.0)]
>>> pairwiseTermPairs u
[(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0)]
>>>  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



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

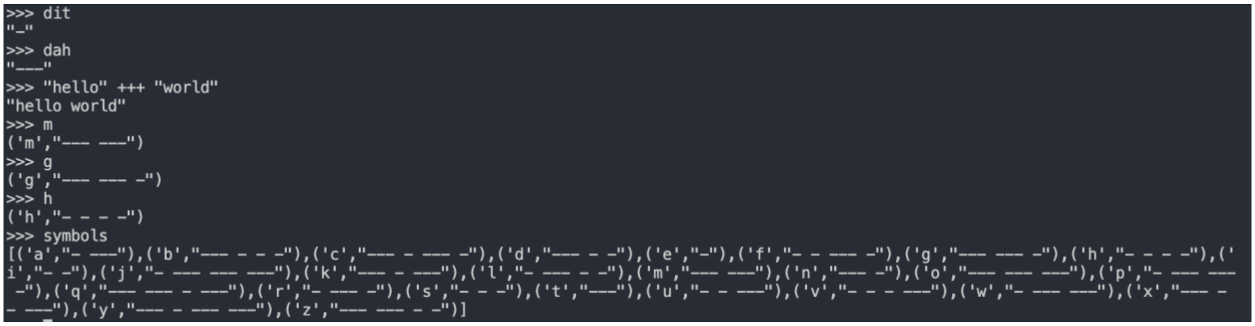
7i



```
>>>
>>> nPVI a
106.34920634920636
>>> nPVI b
88.09523809523809
>>>  nPVI c
37.03703703703703
>>>  nPVI u
0.0
>>>  nPVI x
124.98316498316497
>>>
```

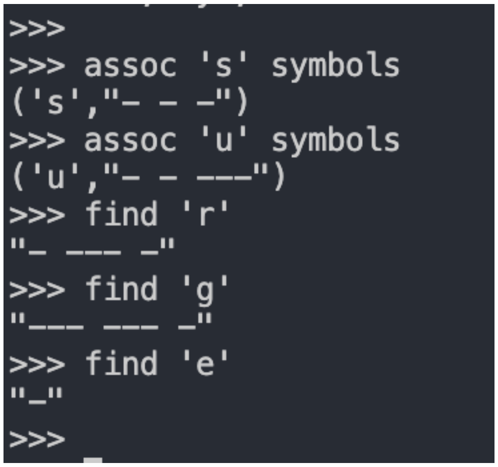
## Task 8- Historic Code: The Dit Dah Code

8a



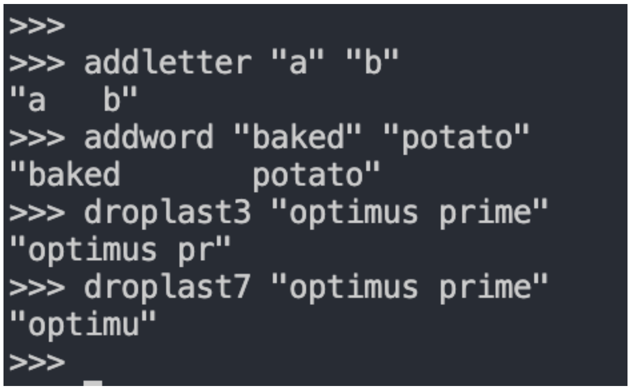
```
>>> dit
"-"
>>> dah
"---"
>>> "hello" +++ "world"
"hello world"
>>> m
('m',"--- ---")
>>> g
('g',"--- --- -")
>>> h
('h',"- - - -")
>>> symbols
[('a',"- ---"),('b',"--- - - -"),('c',"--- - --- -"),('d',"--- - -"),('e',"-"),('f',"- - --- -"),('g',"--- --- -"),('h',"- - - -"),('
i',"- -"),('j',"- --- --- ---"),('k',"--- - ---"),('l',"- --- - -"),('m',"--- ---"),('n',"--- -"),('o',"--- --- ---"),('p',"- --- ---
 -"),('q',"--- --- - ---"),('r',"- --- -"),('s',"- - -"),('t',"---"),('u',"- - ---"),('v',"- - - ---"),('w',"- --- ---"),('x',"--- -
- ---"),('y',"--- - --- ---"),('z',"--- --- - -")]
```

8b



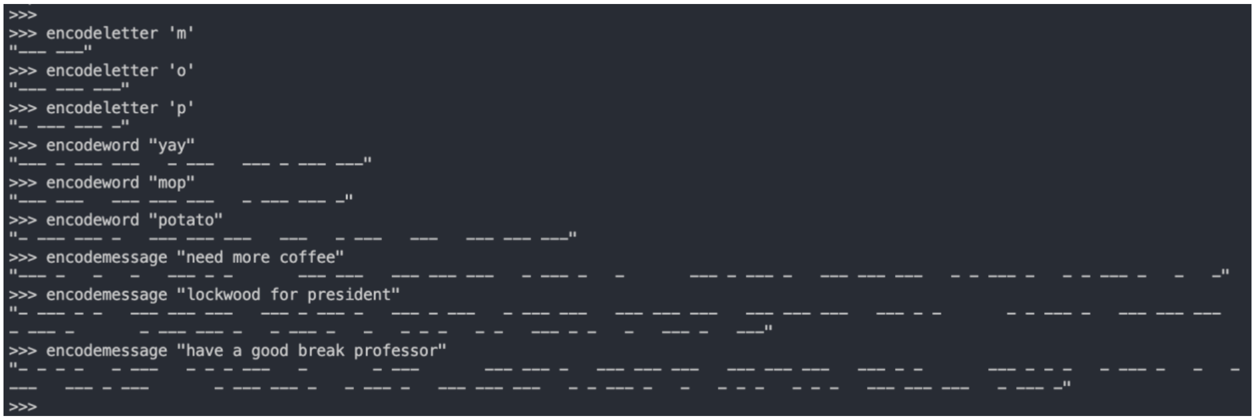
```
>>>
>>> assoc 's' symbols
('s',"- - -")
>>> assoc 'u' symbols
('u',"- - ---")
>>> find 'r'
"- --- -"
>>> find 'g'
"--- --- -"
>>> find 'e'
"-"
>>>
```

8c



```
>>>
>>> addletter "a" "b"
"a   b"
>>> addword "baked" "potato"
"baked       potato"
>>> droplast3 "optimus prime"
"optimus pr"
>>> droplast7 "optimus prime"
"optimu"
>>>
```

8d



```
>>>
>>> encodeletter 'm'
"___ ___"
>>> encodeletter 'o'
"___ ___ ___"
>>> encodeletter 'p'
"_ ___ ___ _"
>>> encodeword "yay"
"___ _ ___ ___   _ ___   ___ _ ___ ___"
>>> encodeword "mop"
"___ ___   ___ ___ ___   _ ___ ___ _"
>>> encodeword "potato"
"_ ___ ___ _   ___ ___ ___   ___   _ ___   ___   ___ ___ ___"
>>> encodemessage "need more coffee"
"___ _   _   _   ___ _ _       ___ ___   ___ ___ ___   _ ___ _   _       ___ _ ___ _   ___ ___ ___   _ _ ___ _   _ _ ___ _   _   _"
>>> encodemessage "lockwood for president"
"_ ___ _ _   ___ ___ ___   ___ _ ___ _   ___ _ ___   _ ___ ___   ___ ___ ___   ___ ___ ___   ___ _ _       _ _ ___ _   ___ ___ ___
_ ___ _       _ ___ ___ _   _ ___ _   _   _ _ _   _ _   ___ _ _   _   ___ _   ___"
>>> encodemessage "have a good break professor"
"_ _ _ _   _ ___   _ _ _ ___   _       _ ___       ___ ___ _   ___ ___ ___   ___ ___ ___   ___ _ _       ___ _ _ _   _ ___ _   _   _
___   ___ _ ___       _ ___ ___ _   _ ___ _   ___ ___ ___   _ _ ___ _   _   _ _ _   _ _ _   ___ ___ ___   _ ___ _"
>>>
```