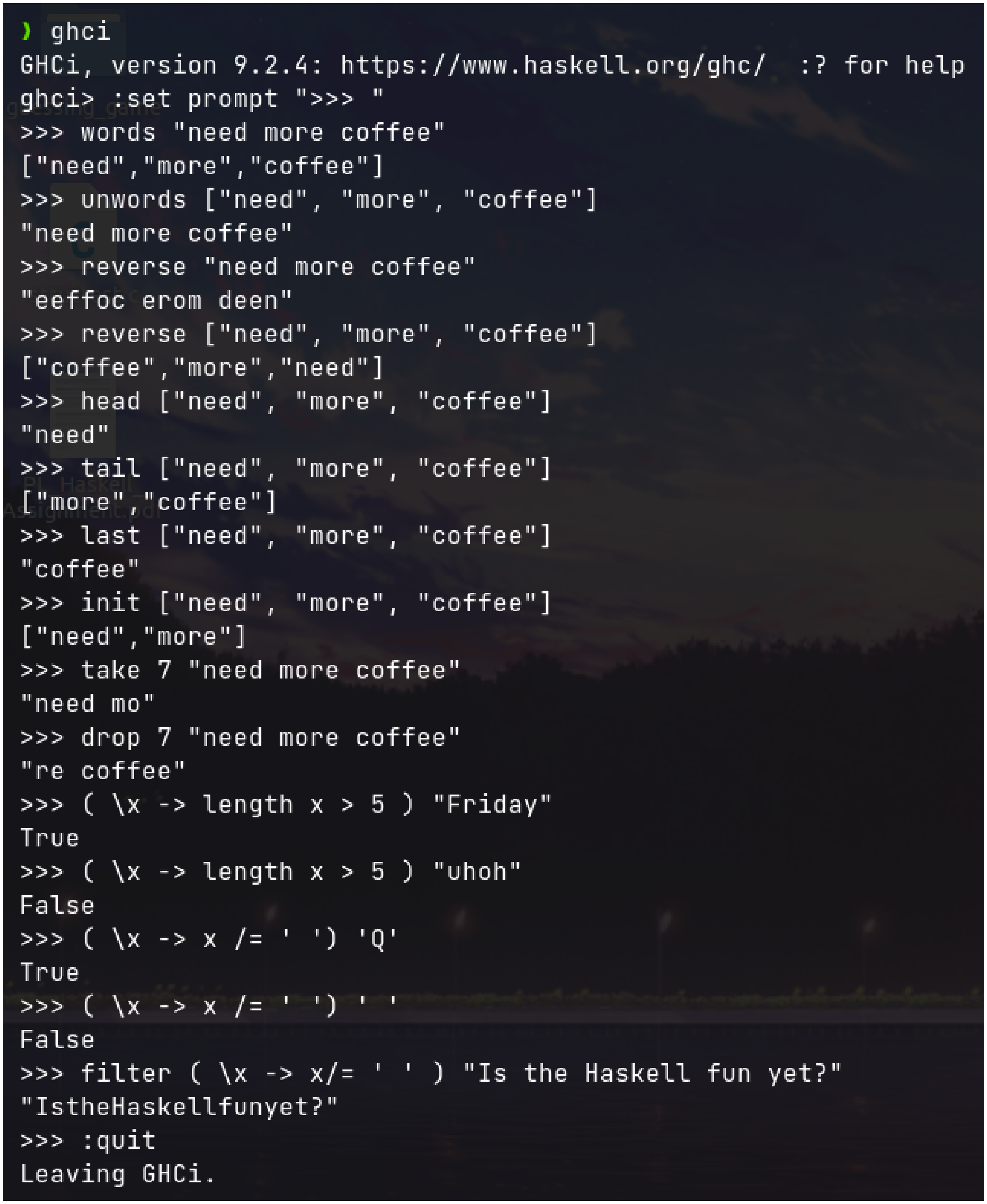
# Haskell Assignment

Eli Fereira

## Abstract

Our first (and only) look at haskell, is a pretty simple, longer assignment, that encapsulates many of the parts of haskell in one document. Brief string processing, algebra, list processing, and higher order functions sections, followed by a large-scale formula, built from scratch using several functions

# Task 1: Mimicking the Demo



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> 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?"
>>> :quit
Leaving GHCi.
```

# Task 2: Numeric Function Definitions

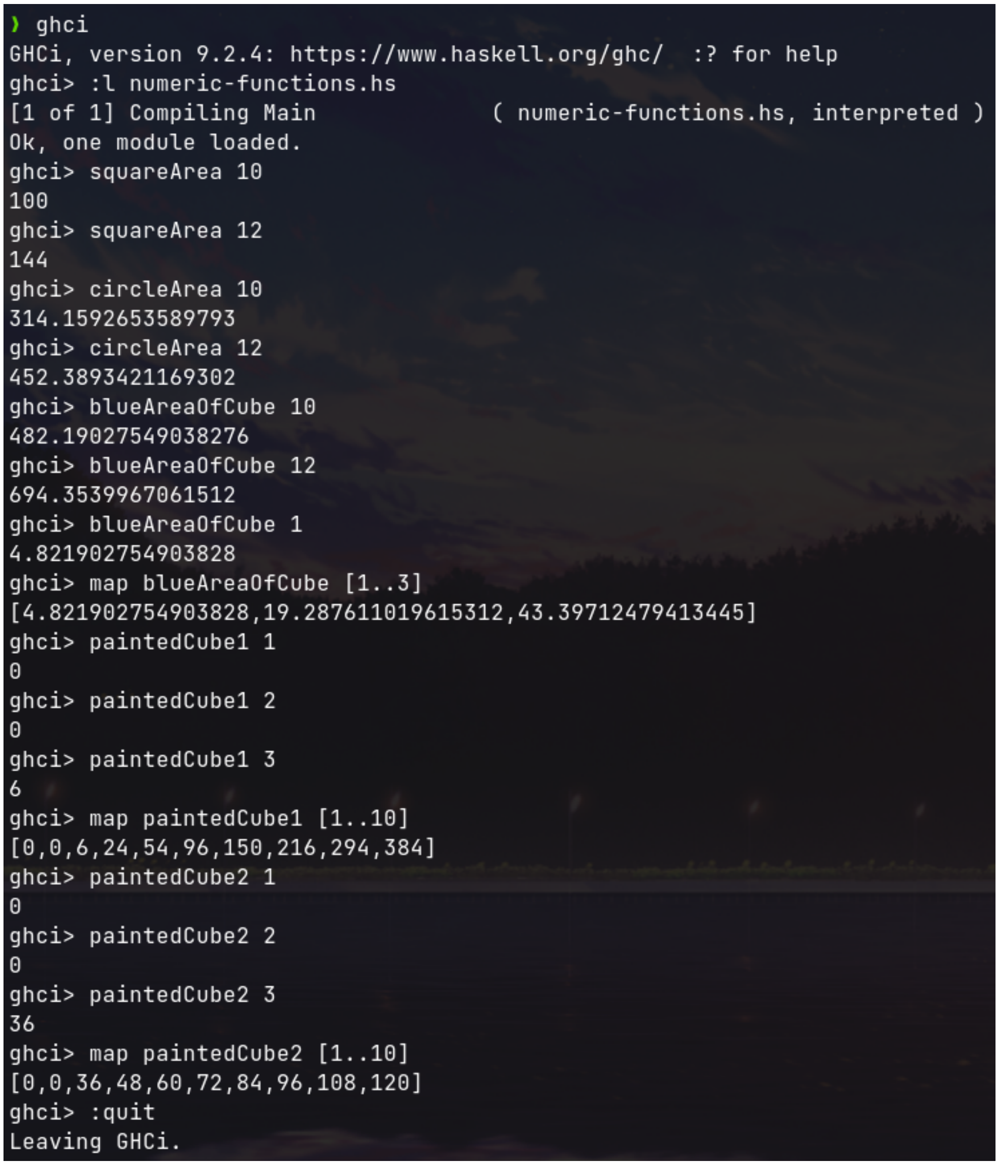
```
squareArea x = x * x

circleArea rad = radsq * pi
    where radsq = rad * rad

blueAreaOfCube side = 6 * ( sarea - carea )
    where sarea = squareArea side
          carea = 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
```

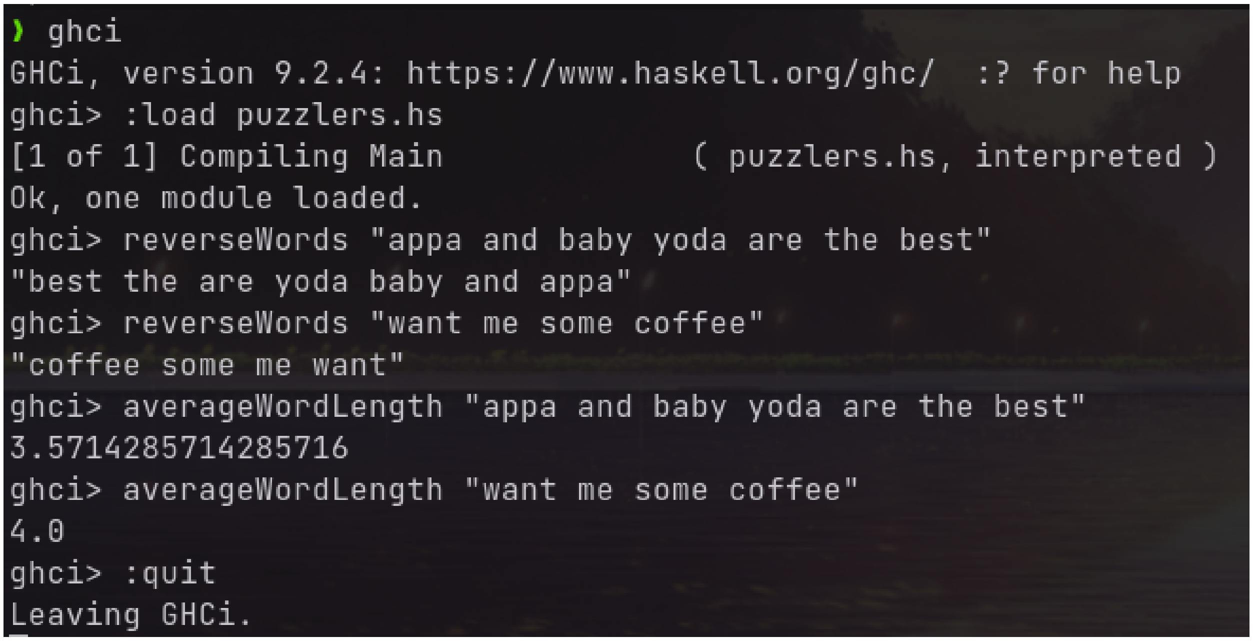


```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l numeric-functions.hs
[1 of 1] Compiling Main             ( numeric-functions.hs, interpreted )
Ok, one module loaded.
ghci> squareArea 10
100
ghci> squareArea 12
144
ghci> circleArea 10
314.1592653589793
ghci> circleArea 12
452.3893421169302
ghci> blueAreaOfCube 10
482.19027549038276
ghci> blueAreaOfCube 12
694.3539967061512
ghci> blueAreaOfCube 1
4.821902754903828
ghci> map blueAreaOfCube [1..3]
[4.821902754903828,19.287611019615312,43.39712479413445]
ghci> paintedCube1 1
0
ghci> paintedCube1 2
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 3
36
ghci> map paintedCube2 [1..10]
[0,0,36,48,60,72,84,96,108,120]
ghci> :quit
Leaving GHCi.
```

# Task 3: Puzzlers

```haskell
reverseWords :: String -> String
reverseWords str =  unwords (reverse (words str))

averageWordLength str =  word_sum / list_length
    where word_sum = fromIntegral(sum ( map length (words str)))
          list_length = fromIntegral(length ( words str ))
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :load puzzlers.hs
[1 of 1] Compiling Main             ( puzzlers.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.
```

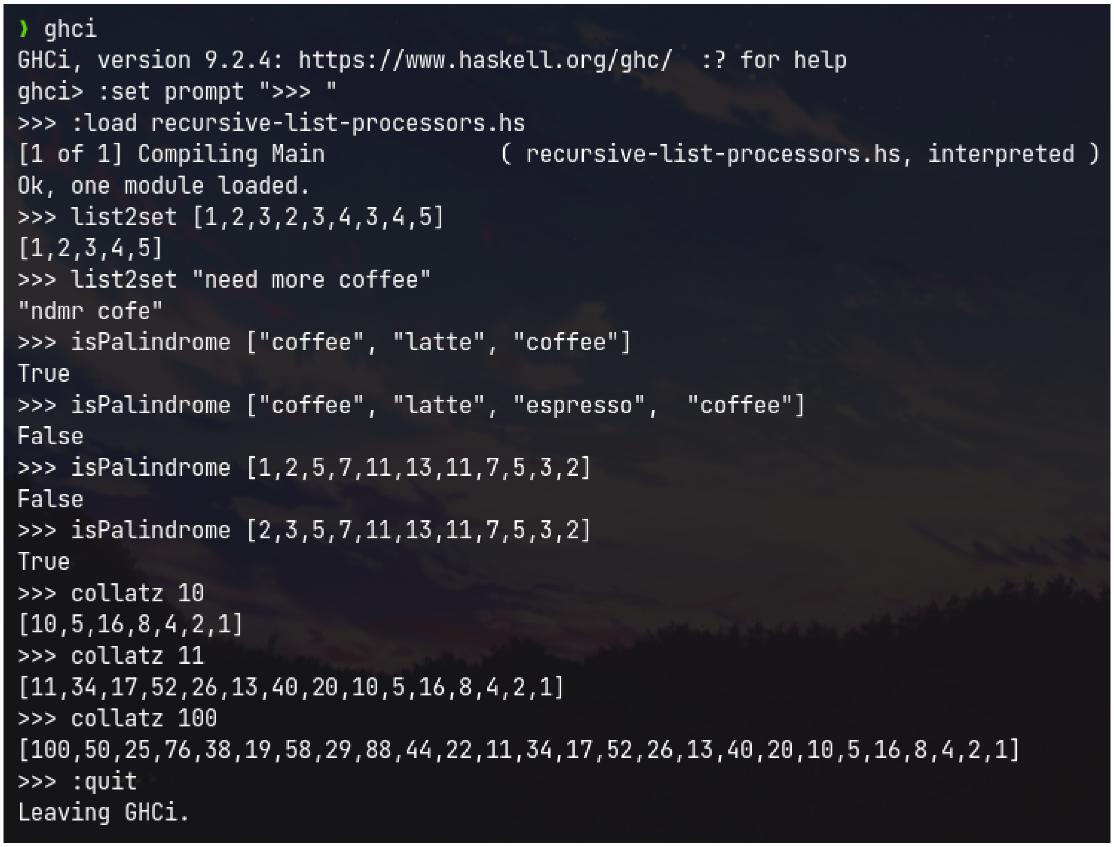
# Task 4: Recursive List Processors

```haskell
list2set [] = []
list2set (el:rest) = if el `elem` rest then list2set rest else el:list2set rest

isPalindrome [] = True
isPalindrome [_] = True
isPalindrome list = head list == last list && isPalindrome innerList
    where innerList = drop 1 endRemovedList
          endRemovedList = reverse ( drop 1 ( reverse list ) )

collatz n = collatzSeq [n]

collatzSeq (1:hist) = reverse (1:hist)
collatzSeq history =
    if even num then collatzSeq (collatzEven:history) else collatzSeq
(collatzOdd:history)
      where num = head history
            collatzEven = div (head history) 2
            collatzOdd = 3 * head history + 1
```

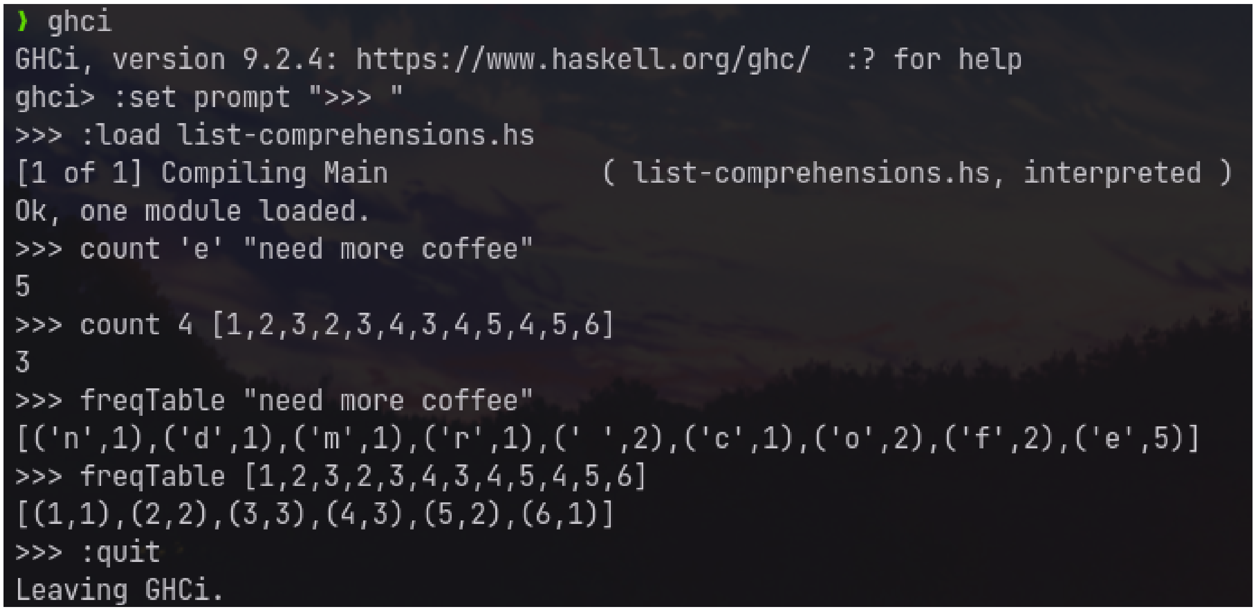


```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load recursive-list-processors.hs
[1 of 1] Compiling Main              ( recursive-list-processors.hs, interpreted )
Ok, one module loaded.
>>> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>> list2set "need more coffee"
"ndmr cofe"
>>> isPalindrome ["coffee", "latte", "coffee"]
True
>>> isPalindrome ["coffee", "latte", "espresso",  "coffee"]
False
>>> 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]
>>> :quit
Leaving GHCi.
```

# Task 5: List Comprehensions

```
count x lx =  sum [ if x == s then 1 else 0 | s <- lx ]

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



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :set prompt ">>> "
>>> :load list-comprehensions.hs
[1 of 1] Compiling Main             ( list-comprehensions.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)]
>>> :quit
Leaving GHCi.
```

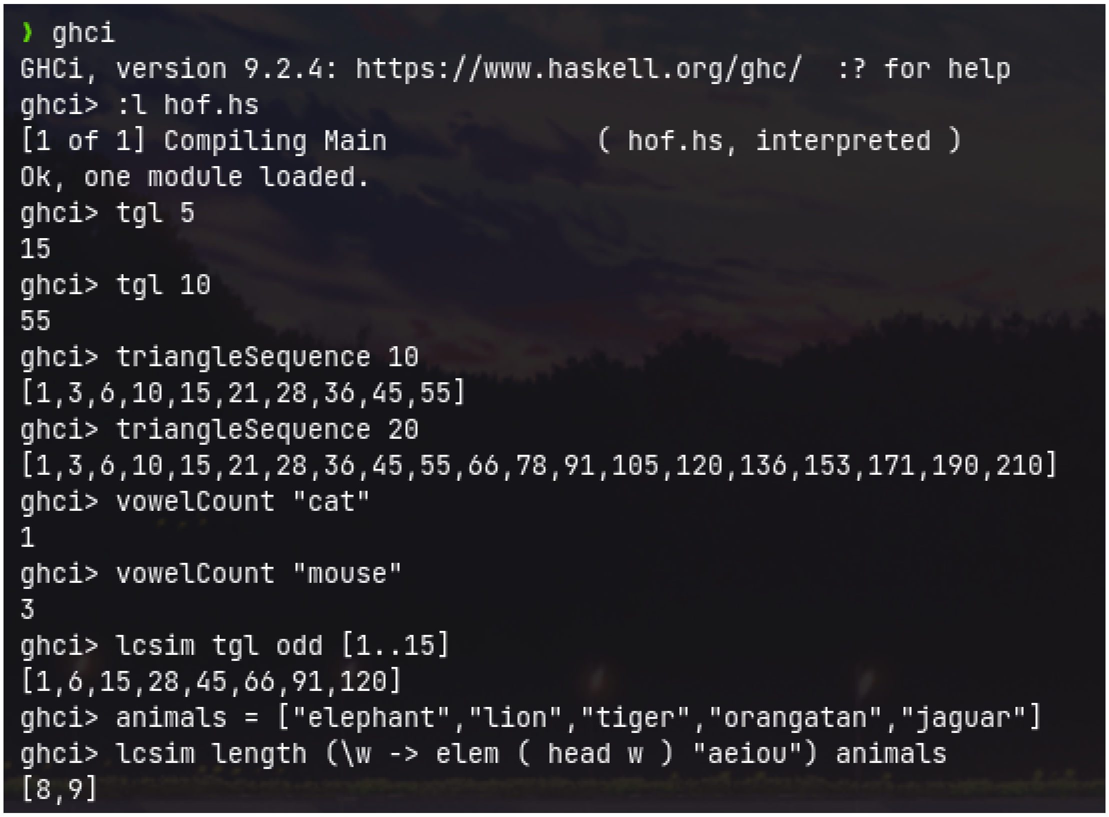
# Task 6: Higher Order Functions

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

triangleSequence n = map tgl [1..n]

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

lcsim mFunc fFunc list = map mFunc ( filter fFunc list )
```

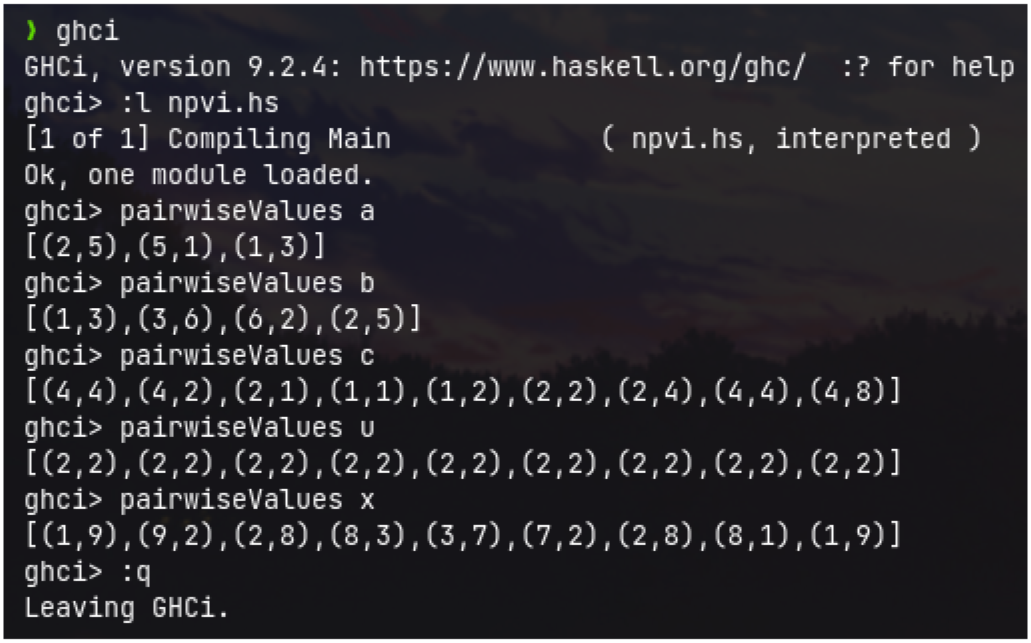


```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l hof.hs
[1 of 1] Compiling Main             ( hof.hs, interpreted )
Ok, one module loaded.
ghci> tgl 5
15
ghci> tgl 10
55
ghci> triangleSequence 10
[1,3,6,10,15,21,28,36,45,55]
ghci> triangleSequence 20
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210]
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]
```

# Task 7: An Interesting Statistic

## Part B

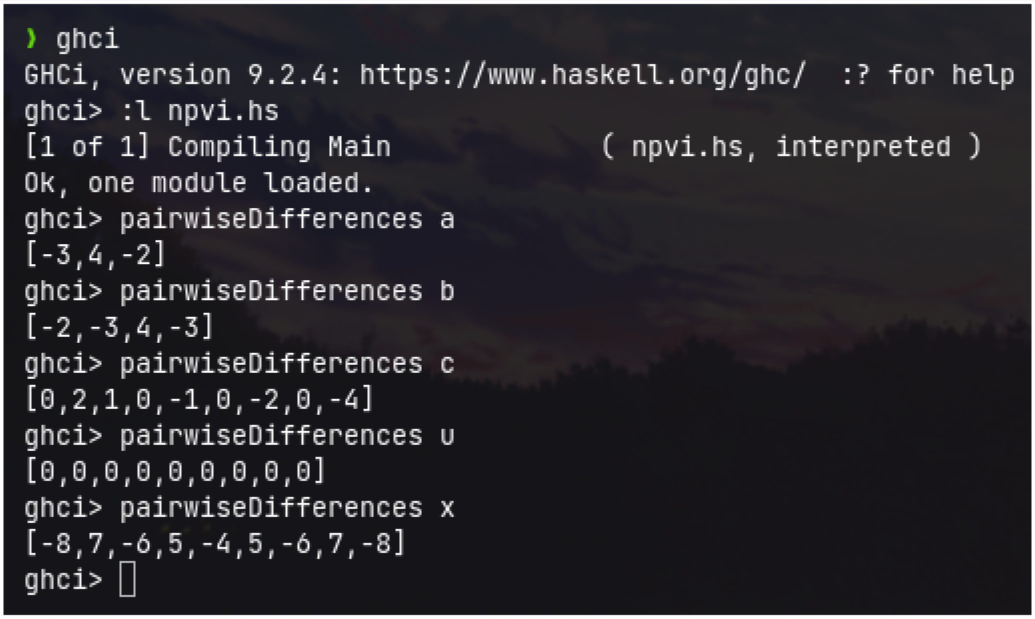
```
pairwiseValues ls = zip ls ( tail ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
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),(4,8)]
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)]
ghci> :q
Leaving GHCi.
```

## Part C

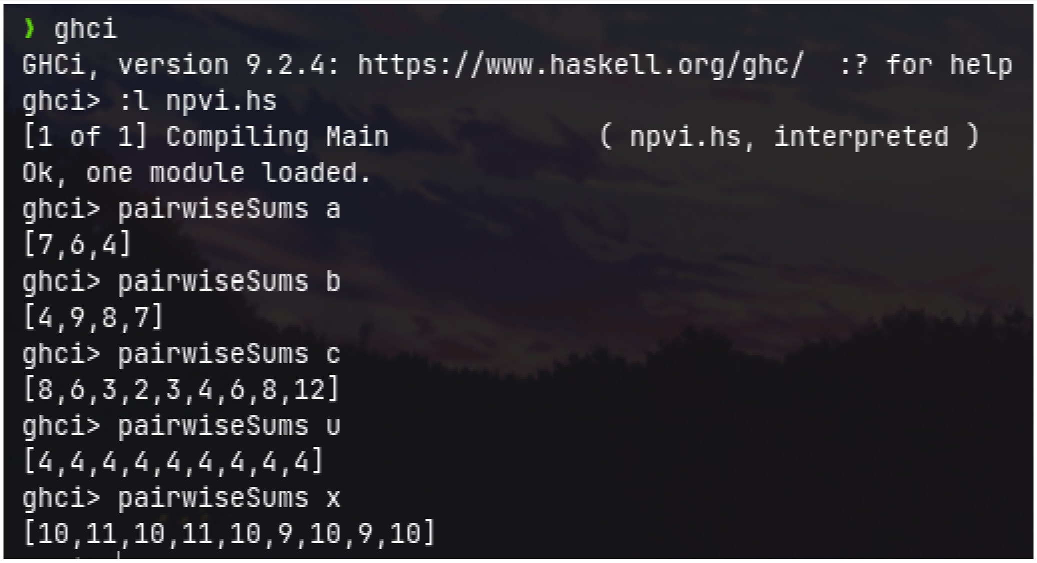
```
pairwiseDifferences ls = map (\ (x,y) -> x - y ) ( pairwiseValues ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> pairwiseDifferences a
[-3,4,-2]
ghci> pairwiseDifferences b
[-2,-3,4,-3]
ghci> pairwiseDifferences c
[0,2,1,0,-1,0,-2,0,-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]
ghci>
```

## Part D

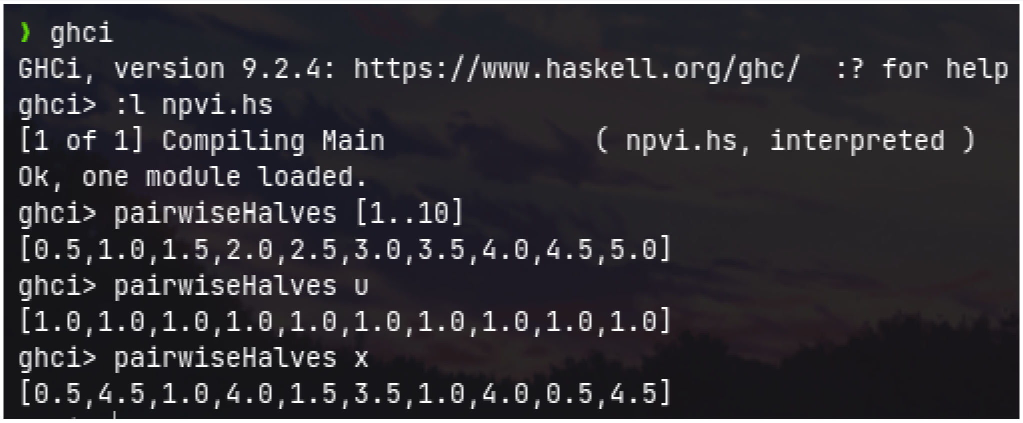
```
pairwiseSums ls = map (\ (x,y) -> x + y ) ( pairwiseValues ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> pairwiseSums a
[7,6,4]
ghci> pairwiseSums b
[4,9,8,7]
ghci> pairwiseSums c
[8,6,3,2,3,4,6,8,12]
ghci> pairwiseSums u
[4,4,4,4,4,4,4,4,4]
ghci> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
```

## Part E

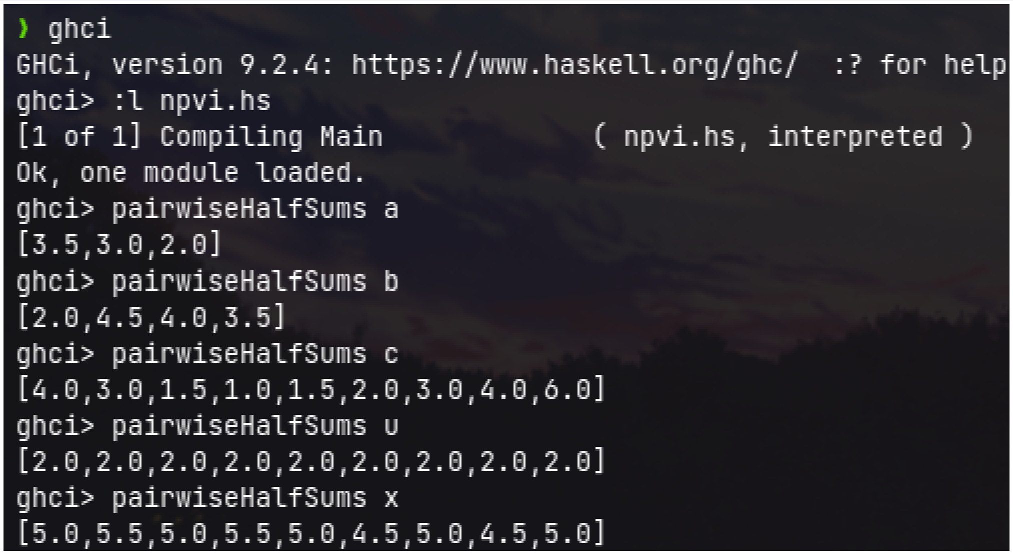
```
half number = fromIntegral number / 2
pairwiseHalves = map half
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
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]
```

## Part F

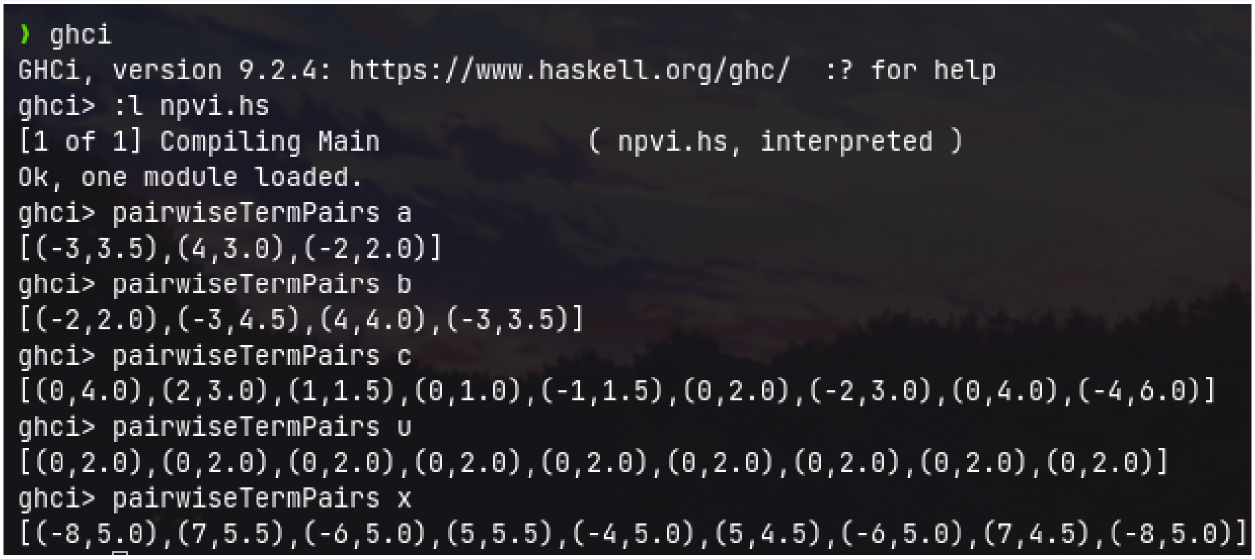
```
pairwiseHalfSums ls = map half ( pairwiseSums ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> pairwiseHalfSums a
[3.5,3.0,2.0]
ghci> pairwiseHalfSums b
[2.0,4.5,4.0,3.5]
ghci> pairwiseHalfSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,6.0]
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]
```

## Part G

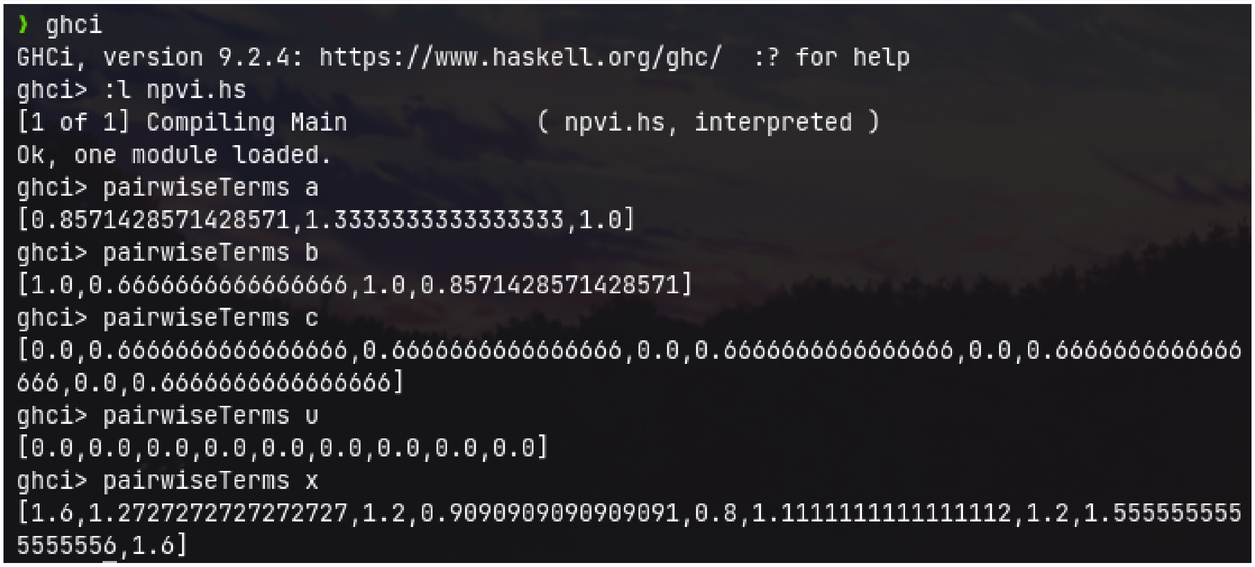
```
pairwiseTermPairs ls = zip ( pairwiseDifferences ls ) ( pairwiseHalfSums ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
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),(-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)]
```

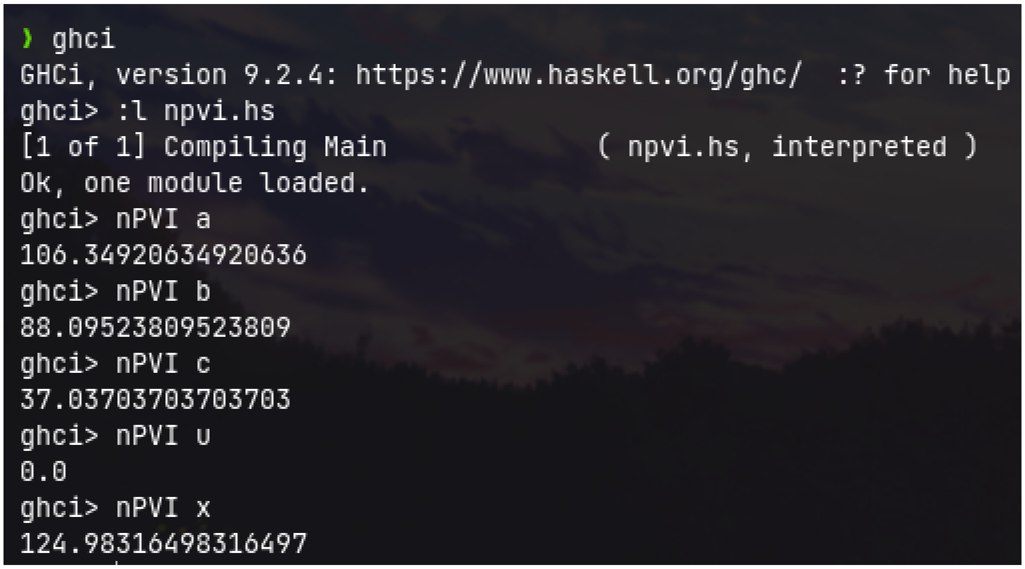
## Part H

```haskell
pairwiseTerms ls = map term ( pairwiseTermPairs ls )
```



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
ghci> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
ghci> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666
666,0.0,0.6666666666666666]
ghci> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
ghci> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.555555555
5555556,1.6]
```

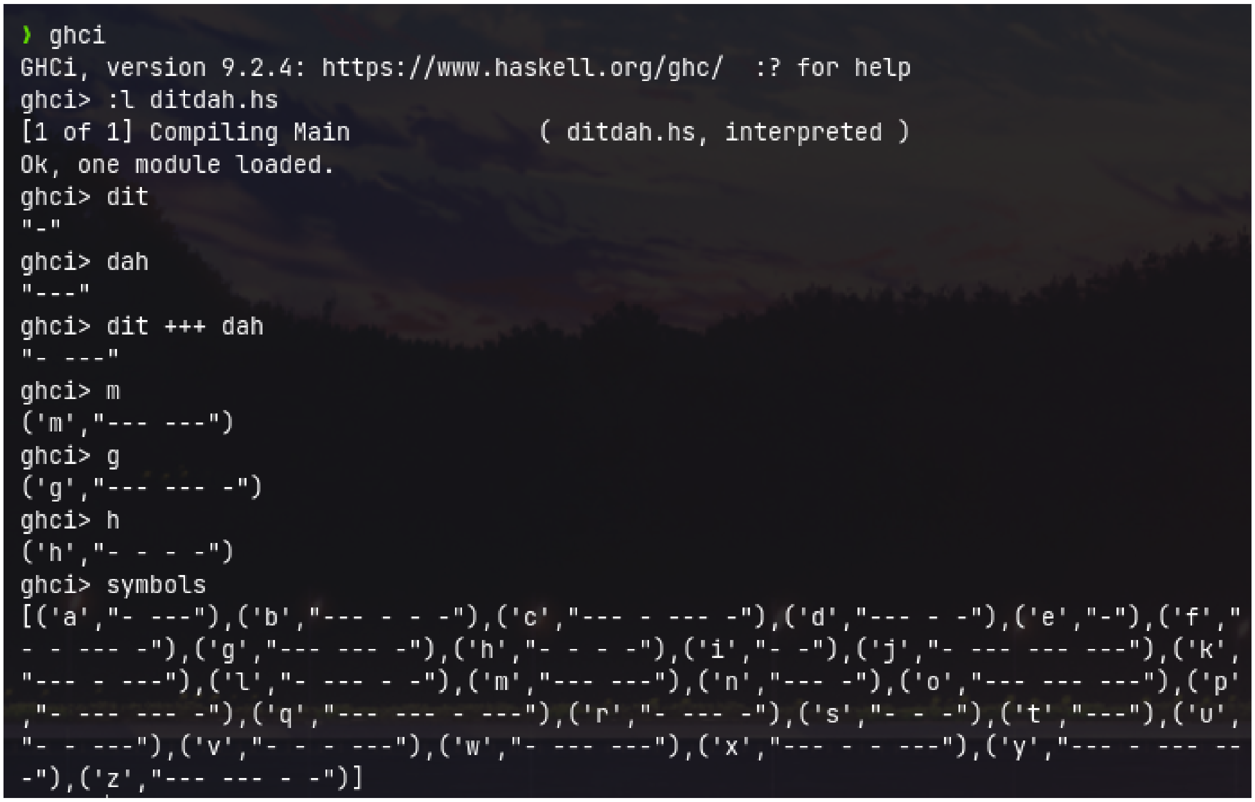
## Part I



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
ghci> nPVI a
106.34920634920636
ghci> nPVI b
88.09523809523809
ghci> nPVI c
37.03703703703703
ghci> nPVI u
0.0
ghci> nPVI x
124.98316498316497
```

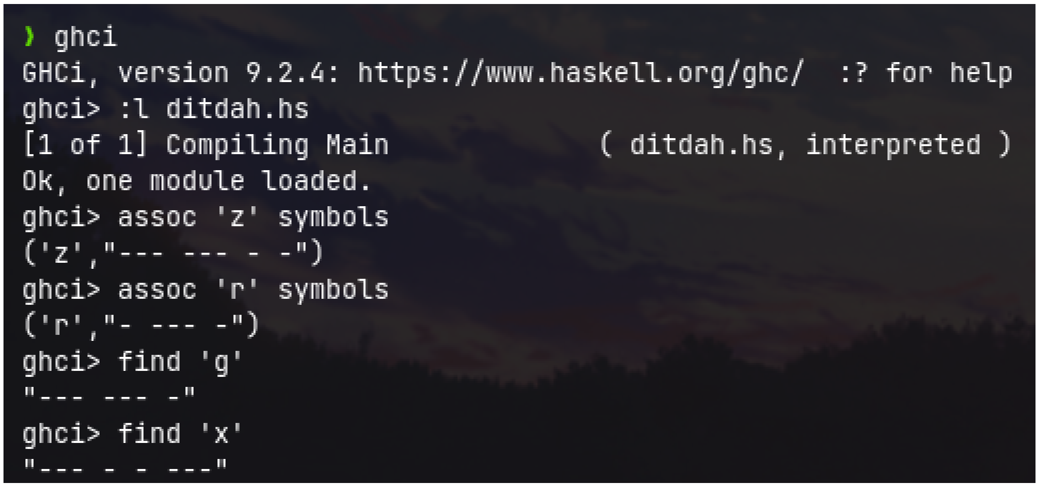
# Task 8: Morse Code

## Part A



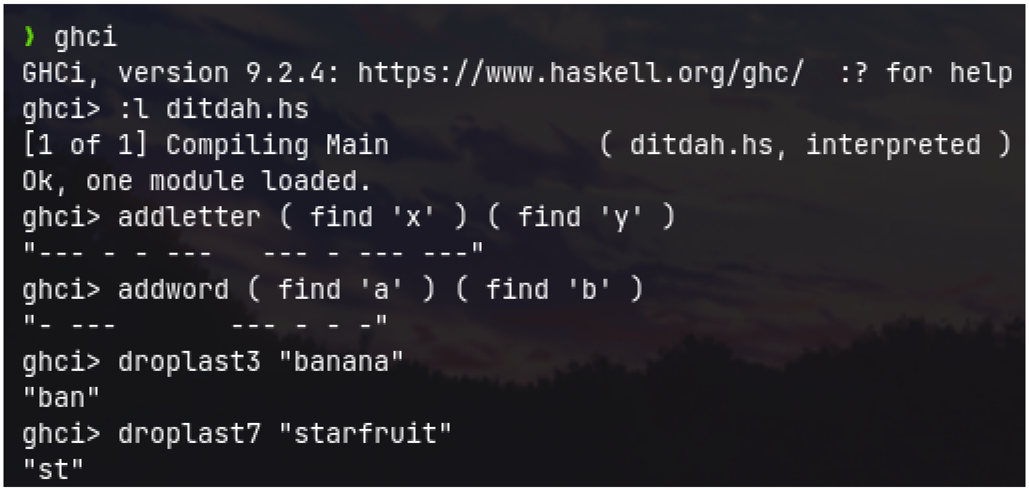
```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> dit
"-"
ghci> dah
"---"
ghci> dit +++ dah
"- ---"
ghci> m
('m',"--- ---")
ghci> g
('g',"--- --- -")
ghci> h
('h',"- - - -")
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',"--- --- - -")]
```

## Part B



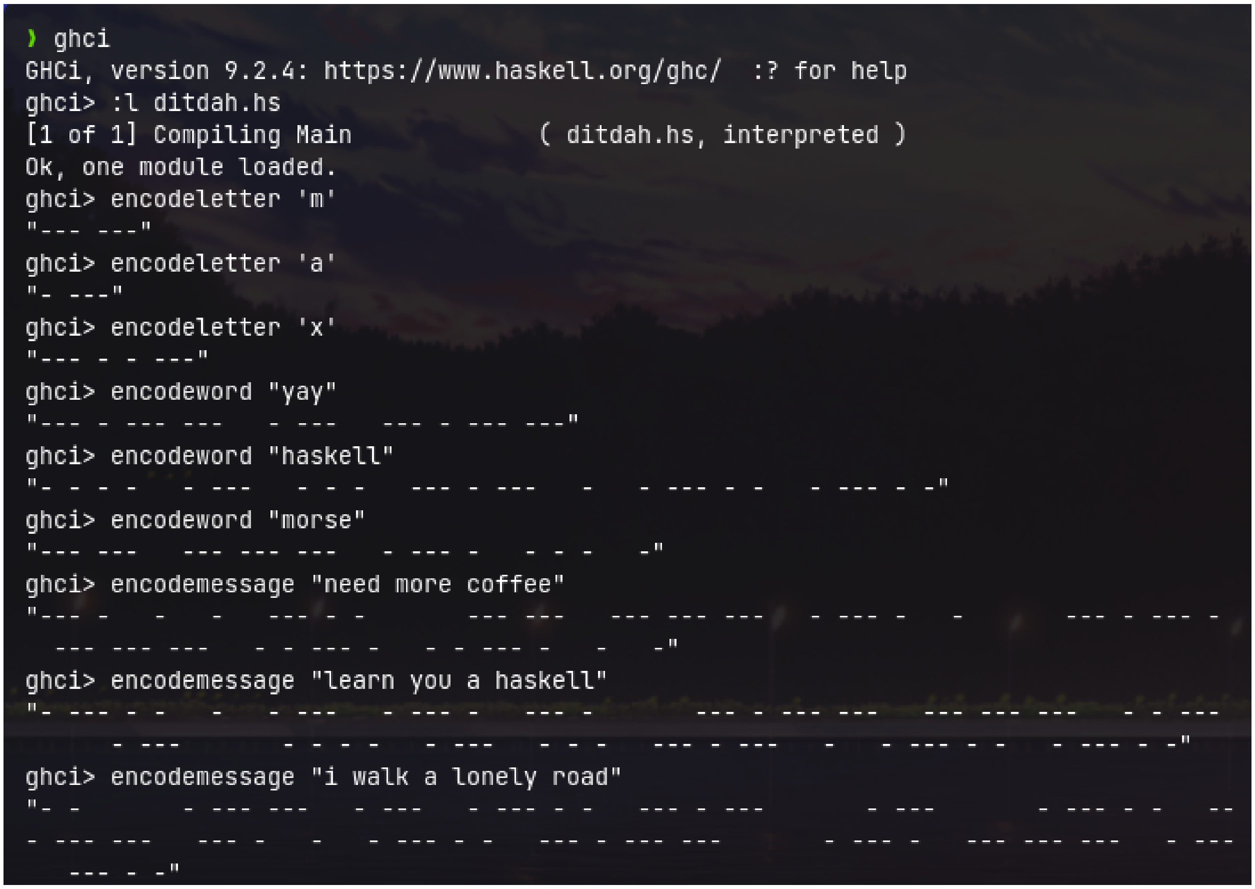
```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> assoc 'z' symbols
('z',"--- --- - -")
ghci> assoc 'r' symbols
('r',"- --- -")
ghci> find 'g'
"--- --- -"
ghci> find 'x'
"--- - - ---"
```

## Part C



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> addletter ( find 'x' ) ( find 'y' )
"--- - - ---   --- - --- ---"
ghci> addword ( find 'a' ) ( find 'b' )
"- ---       --- - - -"
ghci> droplast3 "banana"
"ban"
ghci> droplast7 "starfruit"
"st"
```

## Part D



```
❯ ghci
GHCi, version 9.2.4: https://www.haskell.org/ghc/  :? for help
ghci> :l ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
ghci> encodeletter 'm'
"--- ---"
ghci> encodeletter 'a'
"- ---"
ghci> encodeletter 'x'
"--- - - ---"
ghci> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
ghci> encodeword "haskell"
"- - - -   - ---   - - -   --- - ---   -   - --- - -   - --- - -"
ghci> encodeword "morse"
"--- ---   --- --- ---   - --- -   - - -   -"
ghci> encodemessage "need more coffee"
"--- -   -   -   --- - -       --- ---   --- --- ---   - --- -   -       --- - --- -
  --- --- ---   - - --- -   - - --- -   -   -"
ghci> encodemessage "learn you a haskell"
"- --- - -   -   - ---   - --- -   --- -       --- - --- ---   --- --- ---   - - ---
      - ---       - - - -   - ---   - - -   --- - ---   -   - --- - -   - --- - -"
ghci> encodemessage "i walk a lonely road"
"- -       - --- ---   - ---   - --- - -   --- - ---       - ---       - --- - -   --
- --- ---   --- -   -   - --- - -   --- - --- ---       - --- -   --- --- ---   - ---
   --- - -"
```