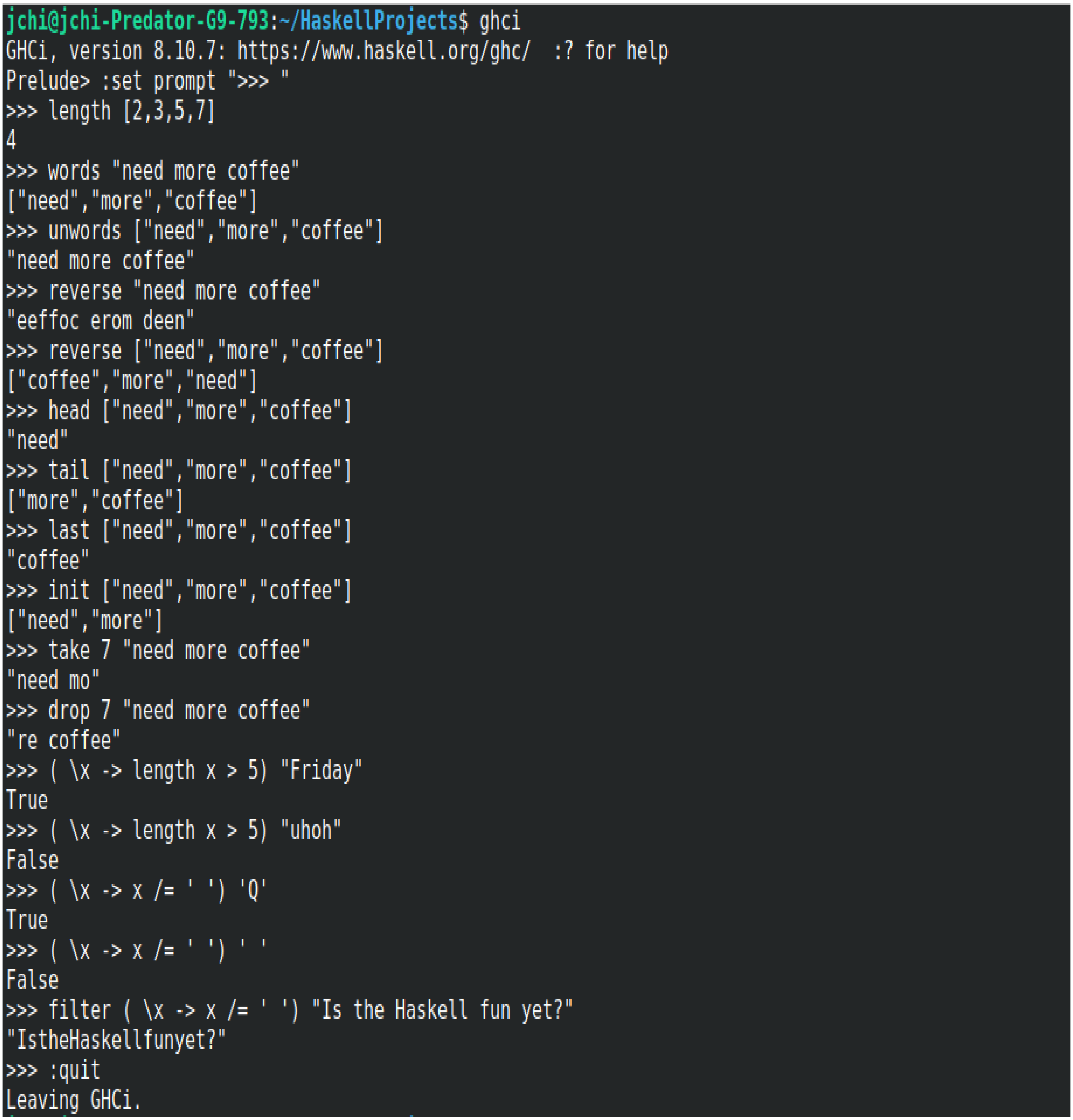
Abstract: In this assignment we were tasked with using implementing Haskell functions in different ways, some include recursion, and list comprehension. We implemented a statistics formula and evaluated morse code.

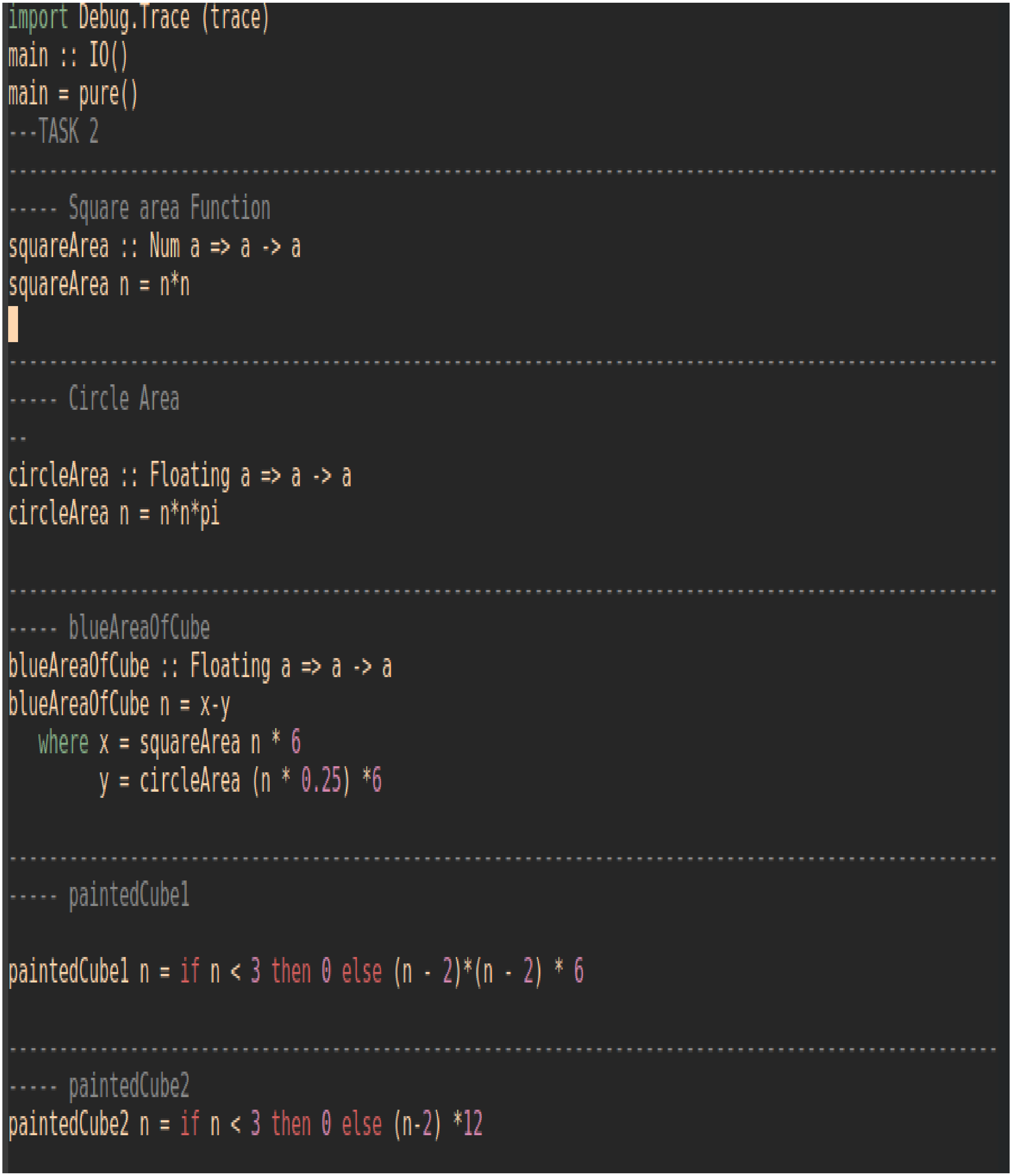
Task 1 - Mindfully Mimicking the Demo



```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :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'
True
>>> ( \x -> x /= ' ') ' '
False
>>> filter ( \x -> x /= ' ') "Is the Haskell fun yet?"
"IstheHaskellfunyet?"
>>> :quit
Leaving GHCi.
```

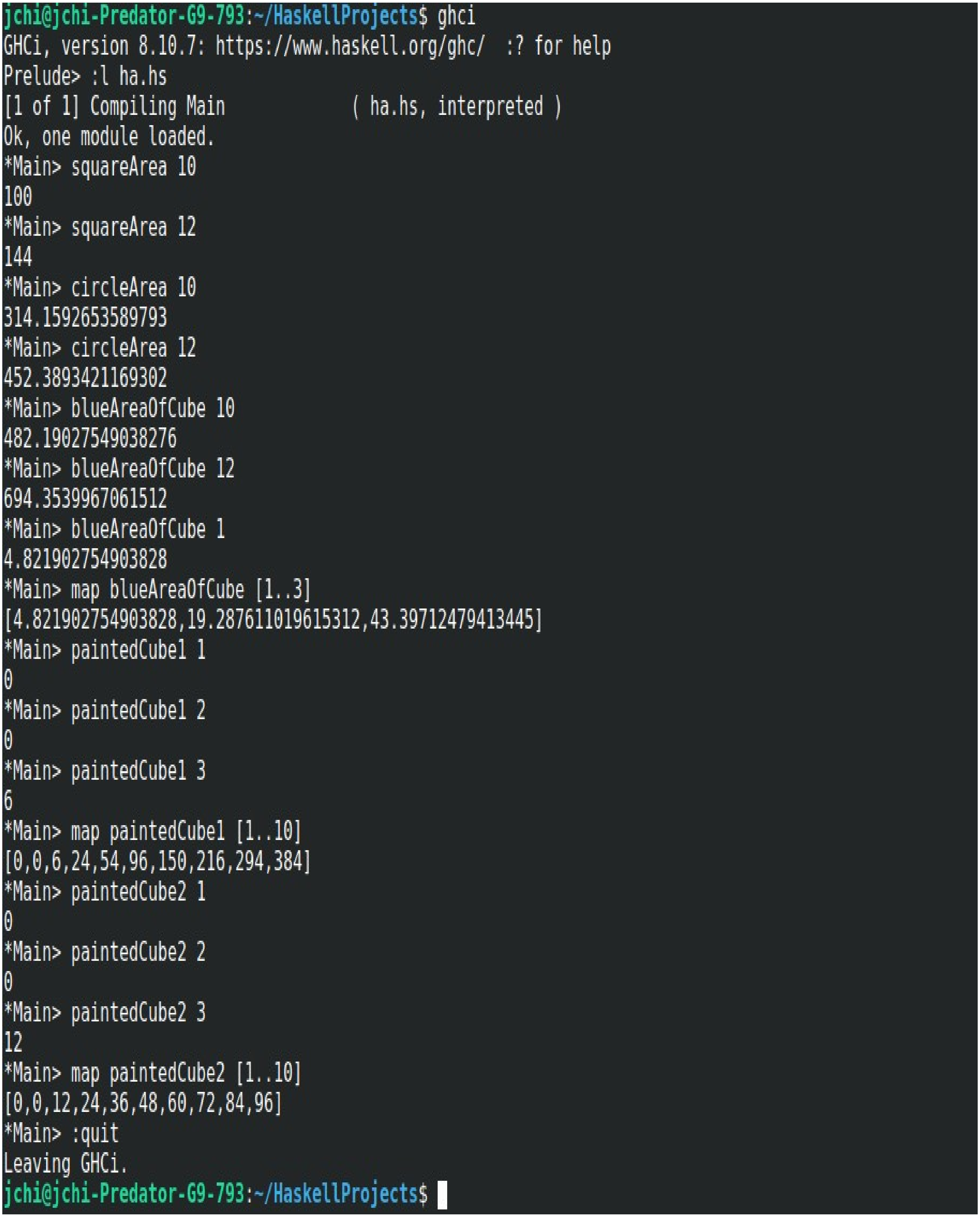
Task 2 - Numeric Function Definitions

Code:



```haskell
import Debug.Trace (trace)
main :: IO()
main = pure()
---TASK 2
--------------------------------------------------------------------------------------------------
----- Square area Function
squareArea :: Num a => a -> a
squareArea n = n*n

--------------------------------------------------------------------------------------------------
----- Circle Area
--
circleArea :: Floating a => a -> a
circleArea n = n*n*pi


--------------------------------------------------------------------------------------------------
----- blueAreaOfCube
blueAreaOfCube :: Floating a => a -> a
blueAreaOfCube n = x-y
   where x = squareArea n * 6
         y = circleArea (n * 0.25) *6


--------------------------------------------------------------------------------------------------
----- paintedCube1

paintedCube1 n = if n < 3 then 0 else (n - 2)*(n - 2) * 6


--------------------------------------------------------------------------------------------------
----- paintedCube2
paintedCube2 n = if n < 3 then 0 else (n-2) *12
```

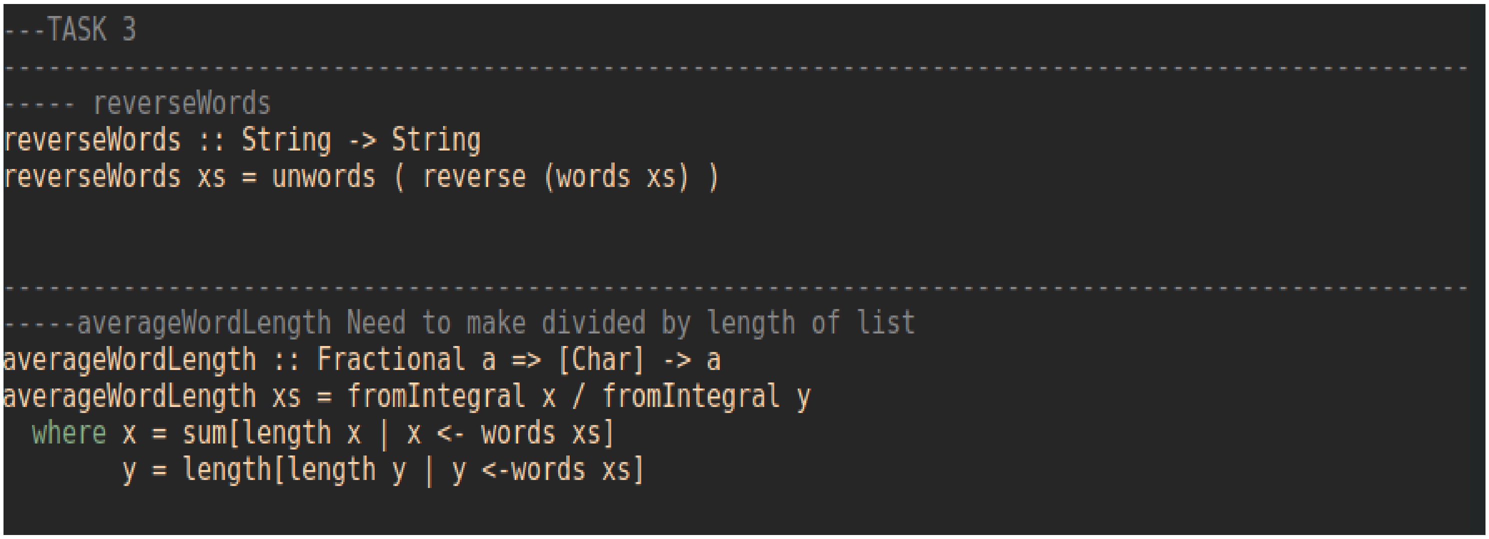
Demo



```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
*Main> squareArea 10
100
*Main> squareArea 12
144
*Main> circleArea 10
314.1592653589793
*Main> circleArea 12
452.3893421169302
*Main> blueAreaOfCube 10
482.19027549038276
*Main> blueAreaOfCube 12
694.3539967061512
*Main> blueAreaOfCube 1
4.821902754903828
*Main> map blueAreaOfCube [1..3]
[4.821902754903828,19.287611019615312,43.39712479413445]
*Main> paintedCube1 1
0
*Main> paintedCube1 2
0
*Main> paintedCube1 3
6
*Main> map paintedCube1 [1..10]
[0,0,6,24,54,96,150,216,294,384]
*Main> paintedCube2 1
0
*Main> paintedCube2 2
0
*Main> paintedCube2 3
12
*Main> map paintedCube2 [1..10]
[0,0,12,24,36,48,60,72,84,96]
*Main> :quit
Leaving GHCi.
jchi@jchi-Predator-G9-793:~/HaskellProjects$
```

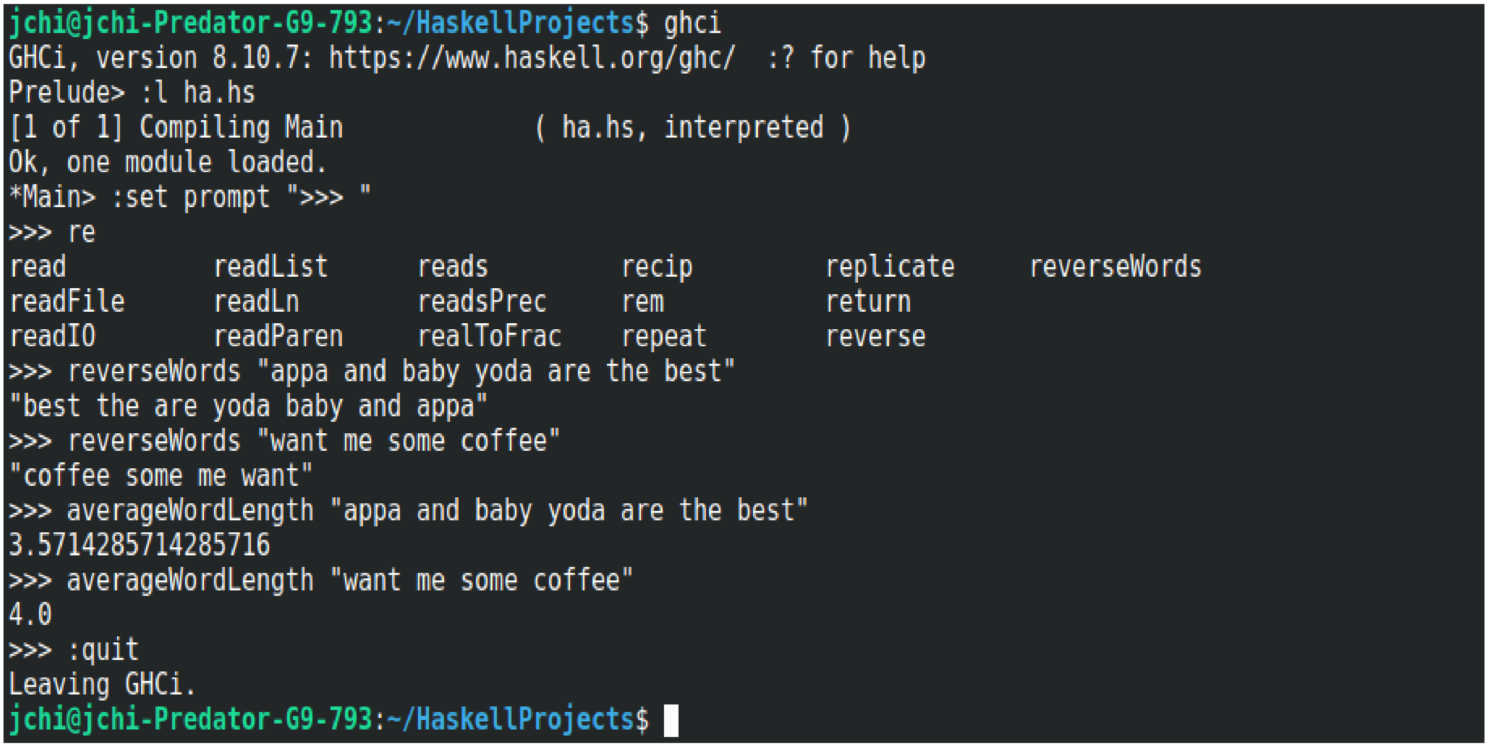
Task 3 - Puzzlers

Code:



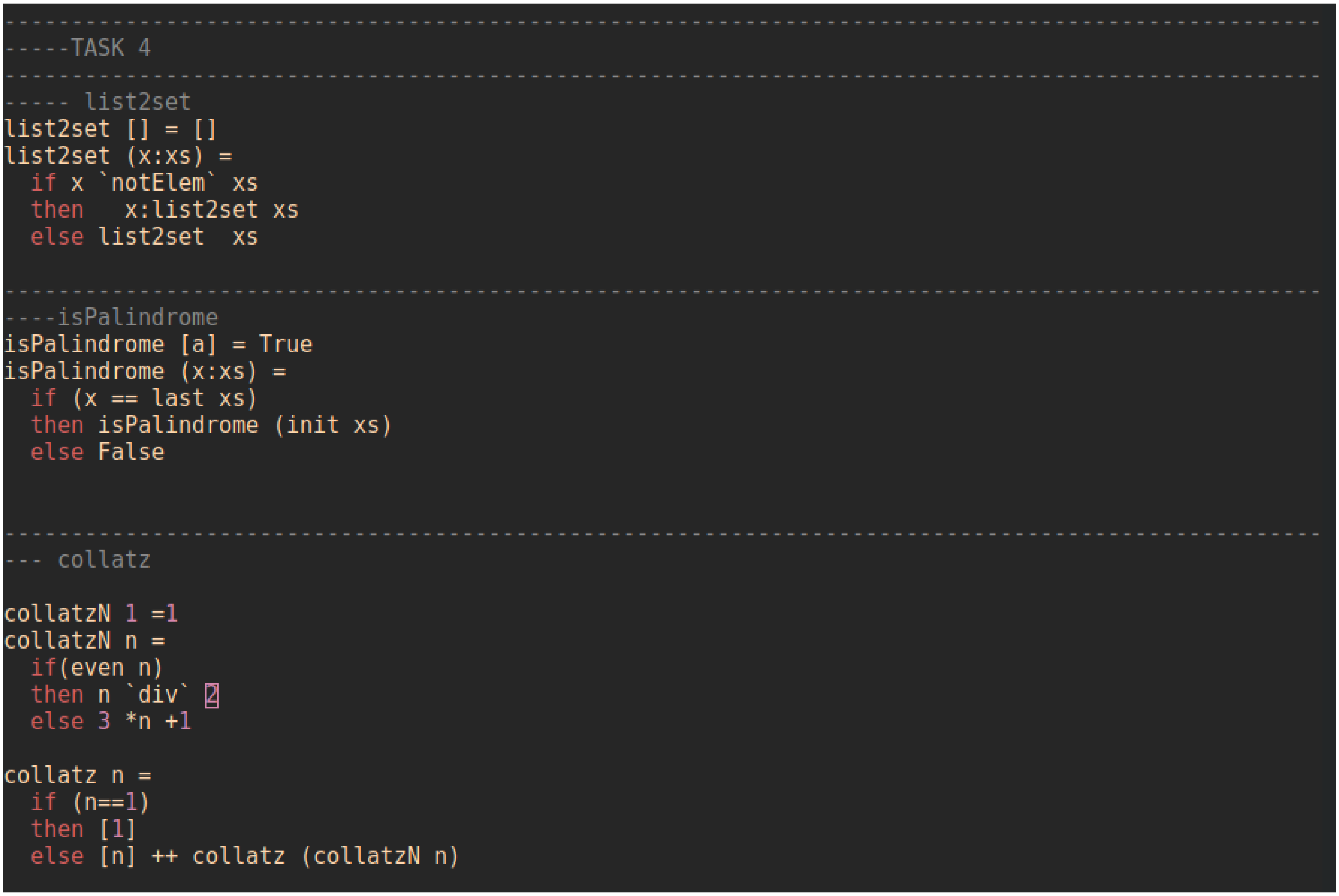
```haskell
---TASK 3
--------------------------------------------------------------------------------------------------
----- reverseWords
reverseWords :: String -> String
reverseWords xs = unwords ( reverse (words xs) )



--------------------------------------------------------------------------------------------------
-----averageWordLength Need to make divided by length of list
averageWordLength :: Fractional a => [Char] -> a
averageWordLength xs = fromIntegral x / fromIntegral y
  where x = sum[length x | x <- words xs]
        y = length[length y | y <-words xs]
```

Demo

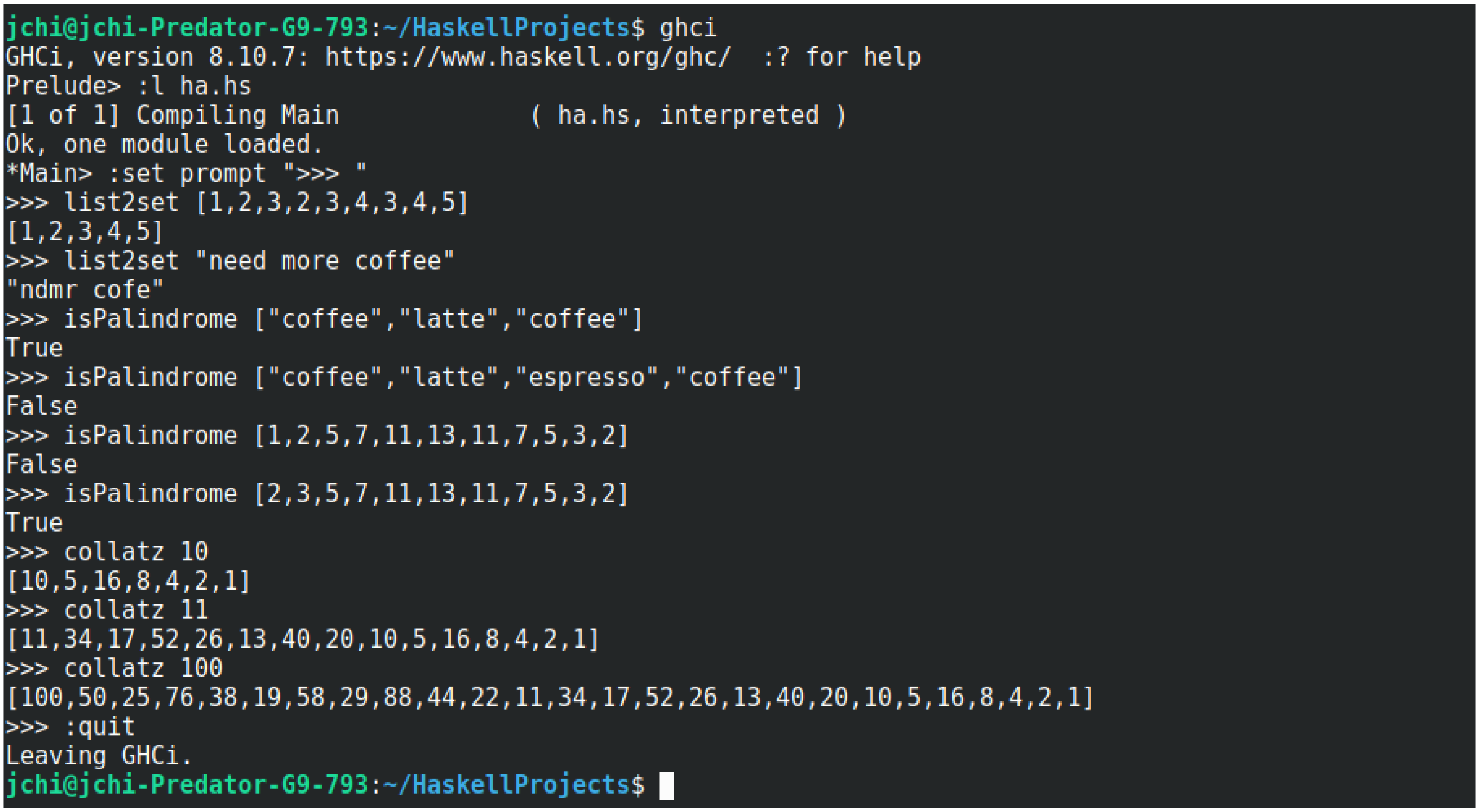


```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
*Main> :set prompt ">>> "
>>> re
read          readList      reads         recip         replicate     reverseWords
readFile      readLn        readsPrec     rem           return
readIO        readParen     realToFrac    repeat        reverse
>>> 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.
jchi@jchi-Predator-G9-793:~/HaskellProjects$ 
```

Task 4 - Recursive List Processors

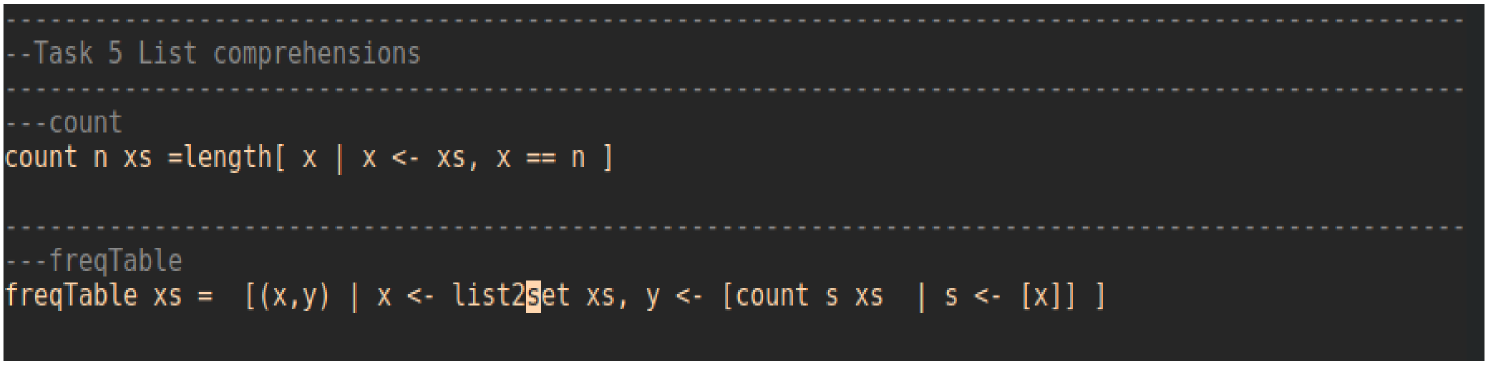


```
---------------------------------------------------------------------------------------------------
-----TASK 4
---------------------------------------------------------------------------------------------------
----- list2set
list2set [] = []
list2set (x:xs) =
  if x `notElem` xs
  then   x:list2set xs
  else list2set  xs

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


---------------------------------------------------------------------------------------------------
--- collatz

collatzN 1 =1
collatzN n =
  if(even n)
  then n `div` 2
  else 3 *n +1

collatz n =
  if (n==1)
  then [1]
  else [n] ++ collatz (collatzN n)
```

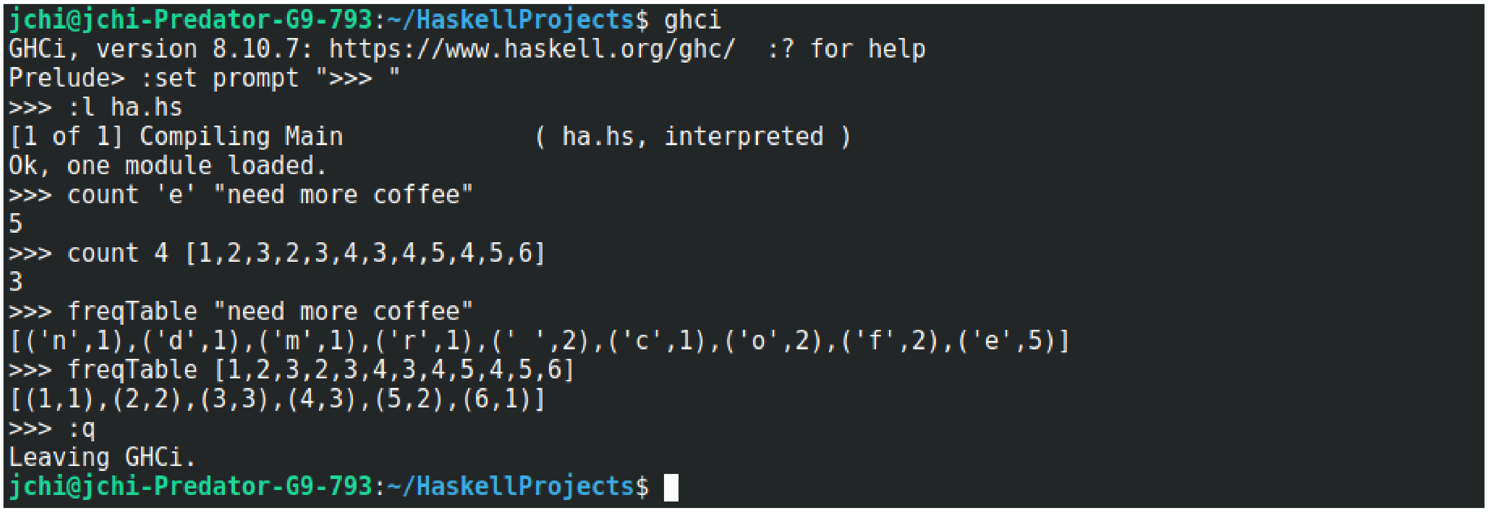


```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
*Main> :set prompt ">>> "
>>> list2set [1,2,3,2,3,4,3,4,5]
[1,2,3,4,5]
>>> list2set "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]
>>> :quit
Leaving GHCi.
jchi@jchi-Predator-G9-793:~/HaskellProjects$ 
```

Task 5 - List Comprehensions

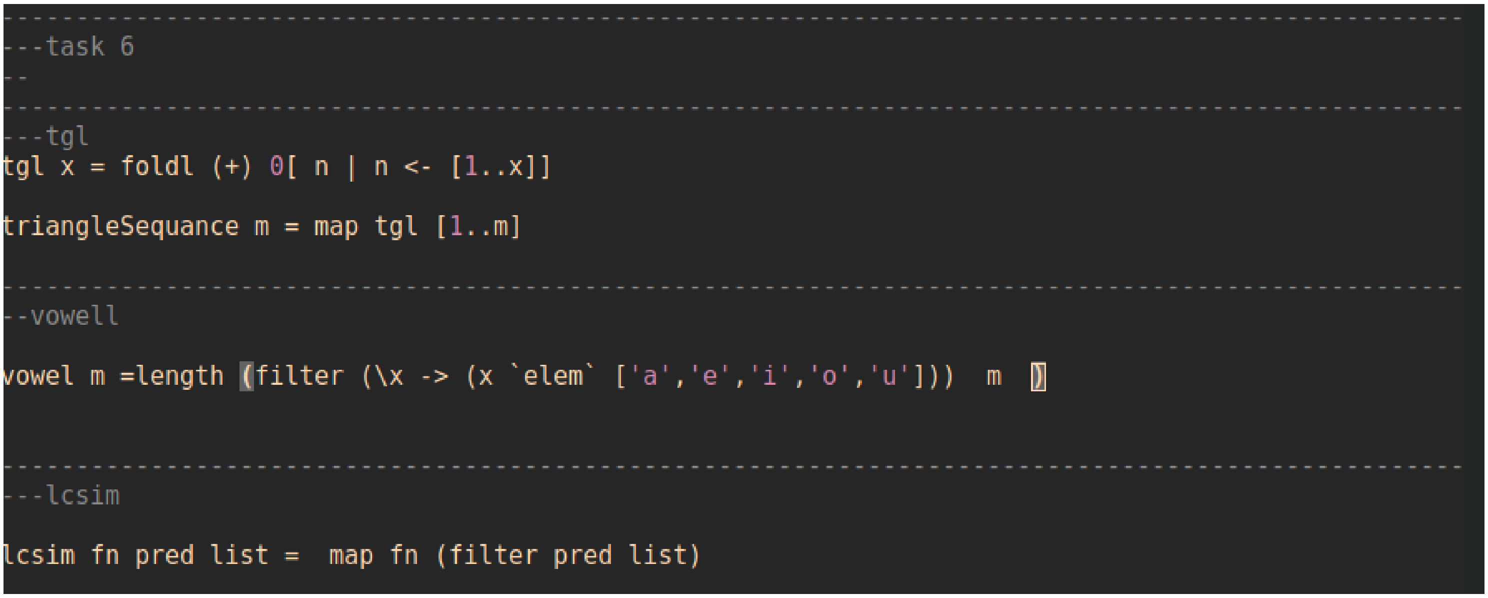


```
-------------------------------------------------------------------------------------------------
--Task 5 List comprehensions
-------------------------------------------------------------------------------------------------
---count
count n xs =length[ x | x <- xs, x == n ]

-------------------------------------------------------------------------------------------------
---freqTable
freqTable xs =  [(x,y) | x <- list2set xs, y <- [count s xs  | s <- [x]] ]
```

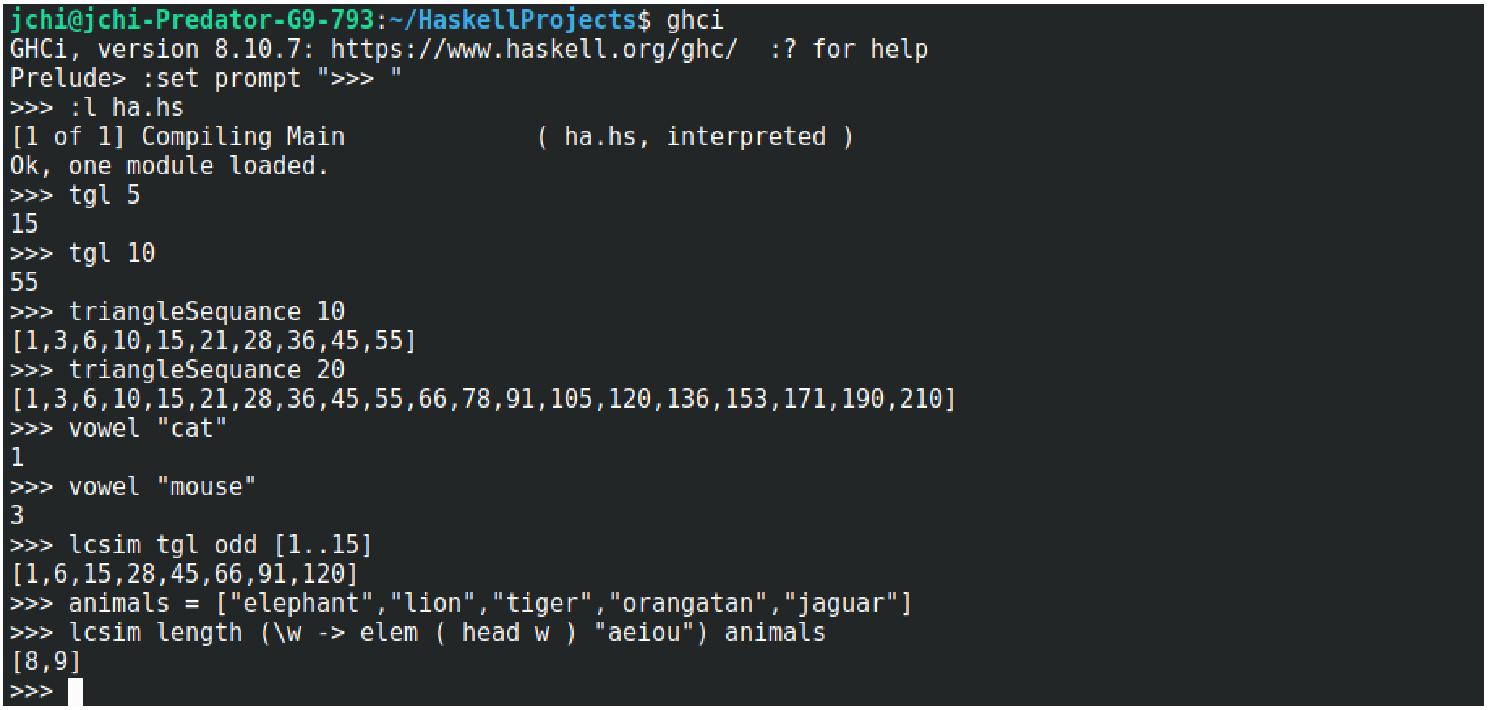


```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> count 'e' "need more coffee"
5
>>> count 4 [1,2,3,2,3,4,3,4,5,4,5,6]
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)]
>>> :q
Leaving GHCi.
jchi@jchi-Predator-G9-793:~/HaskellProjects$ 
```

Task 6 - Higher Order Functions

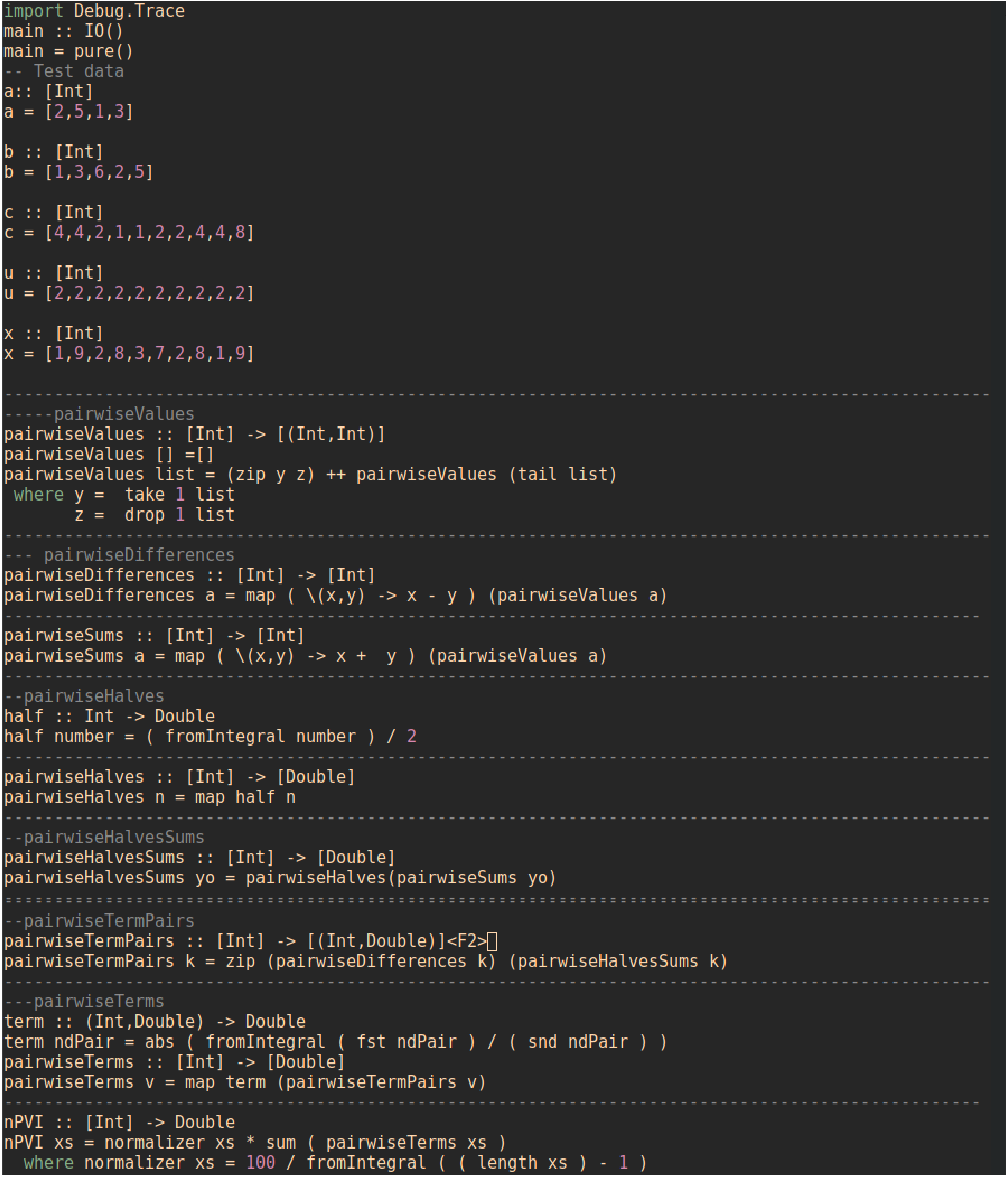


```haskell
--------------------------------------------------------------------------------------------------
---task 6
--
--------------------------------------------------------------------------------------------------
---tgl
tgl x = foldl (+) 0[ n | n <- [1..x]]

triangleSequance m = map tgl [1..m]

--------------------------------------------------------------------------------------------------
--vowell

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


--------------------------------------------------------------------------------------------------
---lcsim

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



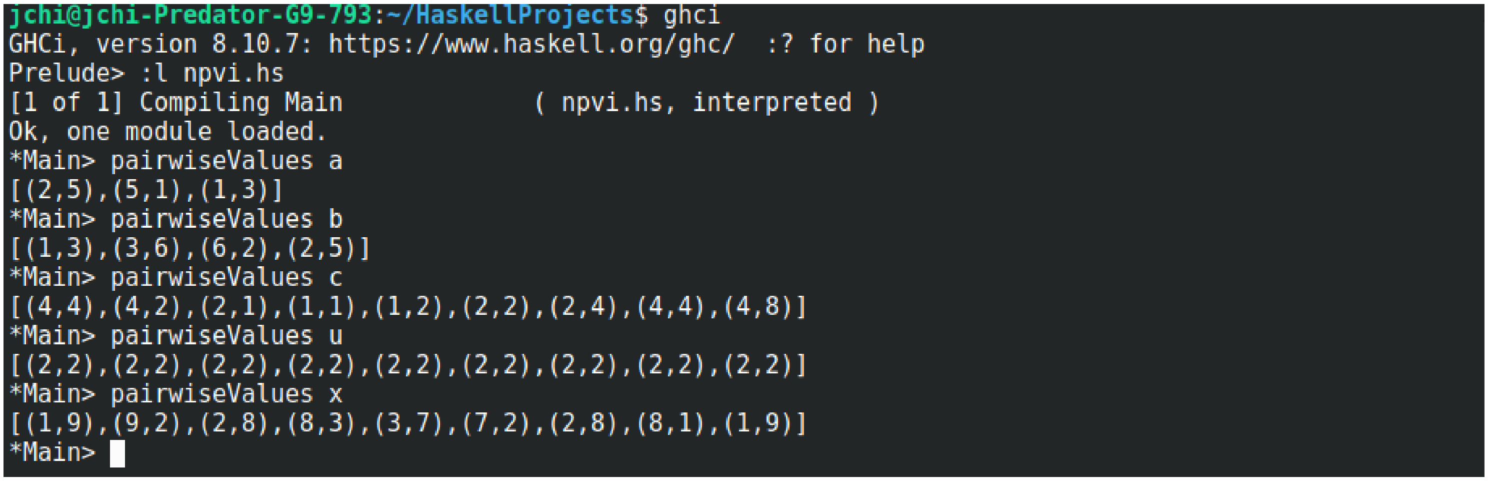
```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :l ha.hs
[1 of 1] Compiling Main             ( ha.hs, interpreted )
Ok, one module loaded.
>>> tgl 5
15
>>> tgl 10
55
>>> triangleSequance 10
[1,3,6,10,15,21,28,36,45,55]
>>> triangleSequance 20
[1,3,6,10,15,21,28,36,45,55,66,78,91,105,120,136,153,171,190,210]
>>> vowel "cat"
1
>>> vowel "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



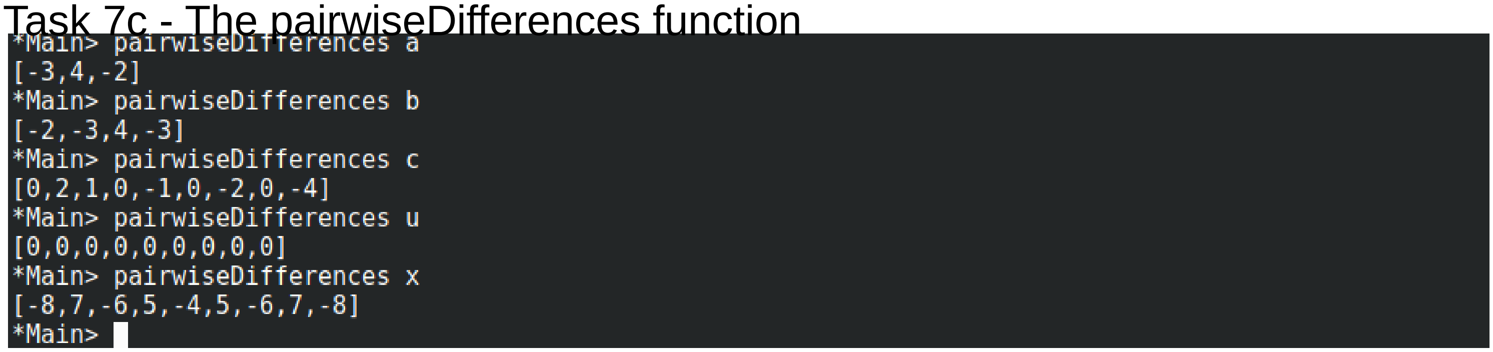
```haskell
import Debug.Trace
main :: IO()
main = pure()
-- Test data
a:: [Int]
a = [2,5,1,3]

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]

--------------------------------------------------------------------------------------------------
-----pairwiseValues
pairwiseValues :: [Int] -> [(Int,Int)]
pairwiseValues [] =[]
pairwiseValues list = (zip y z) ++ pairwiseValues (tail list)
 where y =  take 1 list
       z =  drop 1 list
--------------------------------------------------------------------------------------------------
--- pairwiseDifferences
pairwiseDifferences :: [Int] -> [Int]
pairwiseDifferences a = map ( \(x,y) -> x - y ) (pairwiseValues a)
-------------------------------------------------------------------------------------------------
pairwiseSums :: [Int] -> [Int]
pairwiseSums a = map ( \(x,y) -> x +  y ) (pairwiseValues a)
--------------------------------------------------------------------------------------------------
--pairwiseHalves
half :: Int -> Double
half number = ( fromIntegral number ) / 2
--------------------------------------------------------------------------------------------------
pairwiseHalves :: [Int] -> [Double]
pairwiseHalves n = map half n
--------------------------------------------------------------------------------------------------
--pairwiseHalvesSums
pairwiseHalvesSums :: [Int] -> [Double]
pairwiseHalvesSums yo = pairwiseHalves(pairwiseSums yo)
==================================================================================================
--pairwiseTermPairs
pairwiseTermPairs :: [Int] -> [(Int,Double)]<F2>□
pairwiseTermPairs k = zip (pairwiseDifferences k) (pairwiseHalvesSums k)
--------------------------------------------------------------------------------------------------
---pairwiseTerms
term :: (Int,Double) -> Double
term ndPair = abs ( fromIntegral ( fst ndPair ) / ( snd ndPair ) )
pairwiseTerms :: [Int] -> [Double]
pairwiseTerms v = map term (pairwiseTermPairs v)
-------------------------------------------------------------------------------------------------
nPVI :: [Int] -> Double
nPVI xs = normalizer xs * sum ( pairwiseTerms xs )
  where normalizer xs = 100 / fromIntegral ( ( length xs ) - 1 )
```

Task 7b - The pairwiseValues function



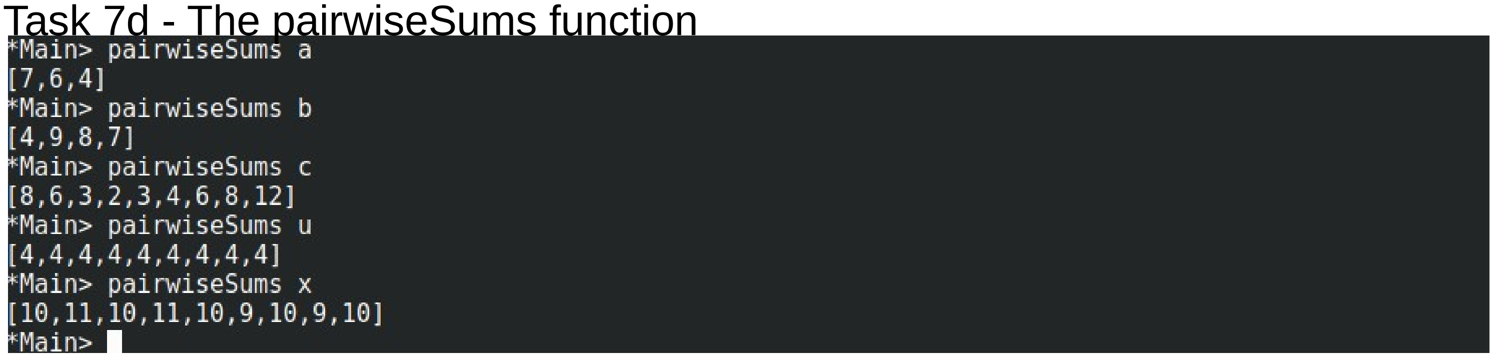
```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :l npvi.hs
[1 of 1] Compiling Main             ( npvi.hs, interpreted )
Ok, one module loaded.
*Main> pairwiseValues a
[(2,5),(5,1),(1,3)]
*Main> pairwiseValues b
[(1,3),(3,6),(6,2),(2,5)]
*Main> pairwiseValues c
[(4,4),(4,2),(2,1),(1,1),(1,2),(2,2),(2,4),(4,4),(4,8)]
*Main> pairwiseValues u
[(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2),(2,2)]
*Main> pairwiseValues x
[(1,9),(9,2),(2,8),(8,3),(3,7),(7,2),(2,8),(8,1),(1,9)]
*Main> 
```

Task 7c - The pairwiseDifferences function



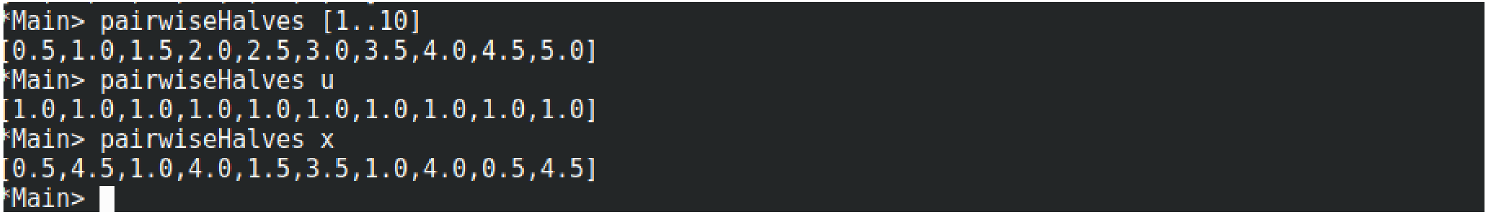
```
*Main> pairwiseDifferences a
[-3,4,-2]
*Main> pairwiseDifferences b
[-2,-3,4,-3]
*Main> pairwiseDifferences c
[0,2,1,0,-1,0,-2,0,-4]
*Main> pairwiseDifferences u
[0,0,0,0,0,0,0,0,0]
*Main> pairwiseDifferences x
[-8,7,-6,5,-4,5,-6,7,-8]
*Main> 
```

Task 7d - The pairwiseSums function



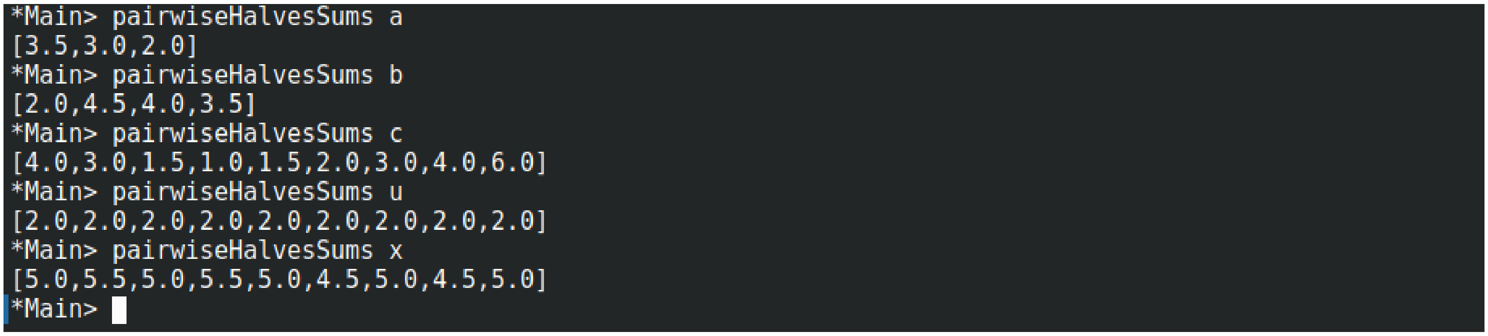
```
*Main> pairwiseSums a
[7,6,4]
*Main> pairwiseSums b
[4,9,8,7]
*Main> pairwiseSums c
[8,6,3,2,3,4,6,8,12]
*Main> pairwiseSums u
[4,4,4,4,4,4,4,4,4]
*Main> pairwiseSums x
[10,11,10,11,10,9,10,9,10]
*Main> 
```

Task 7e - The pairwiseHalves function



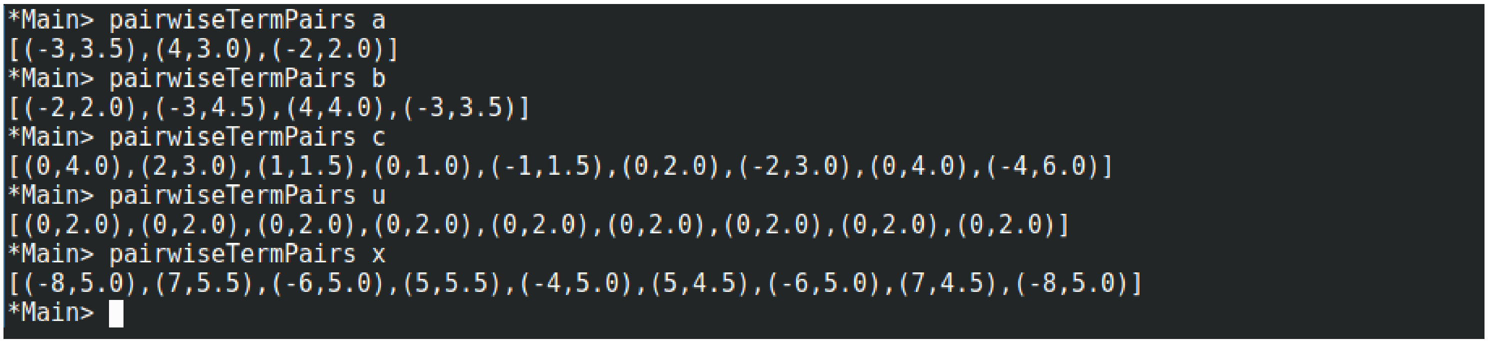
```
*Main> pairwiseHalves [1..10]
[0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
*Main> pairwiseHalves u
[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
*Main> pairwiseHalves x
[0.5,4.5,1.0,4.0,1.5,3.5,1.0,4.0,0.5,4.5]
*Main> 
```

Task 7f - The pairwiseHalfSums function



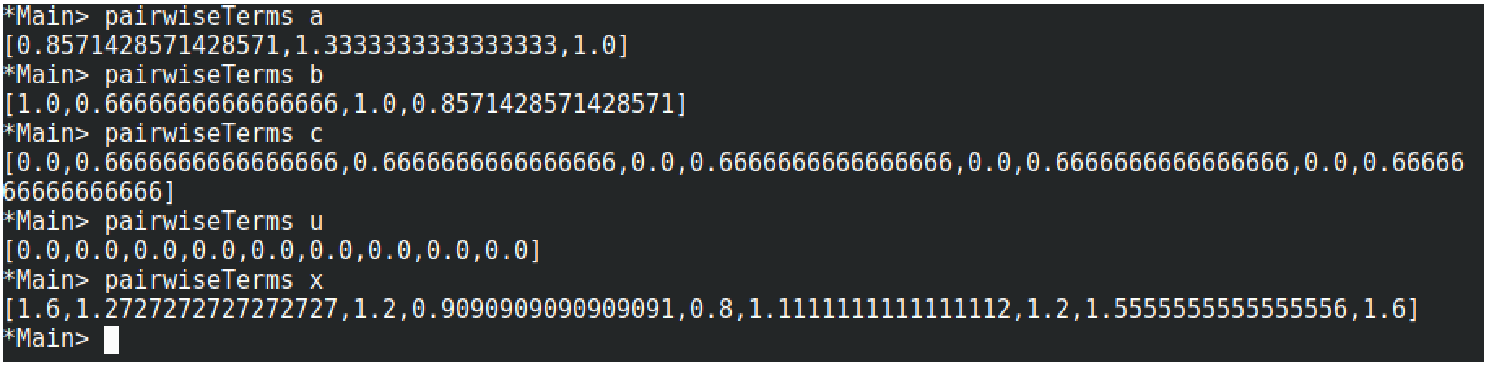
```
*Main> pairwiseHalvesSums a
[3.5,3.0,2.0]
*Main> pairwiseHalvesSums b
[2.0,4.5,4.0,3.5]
*Main> pairwiseHalvesSums c
[4.0,3.0,1.5,1.0,1.5,2.0,3.0,4.0,6.0]
*Main> pairwiseHalvesSums u
[2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0]
*Main> pairwiseHalvesSums x
[5.0,5.5,5.0,5.5,5.0,4.5,5.0,4.5,5.0]
*Main> 
```

Task 7g - The pairwiseTermPairs function



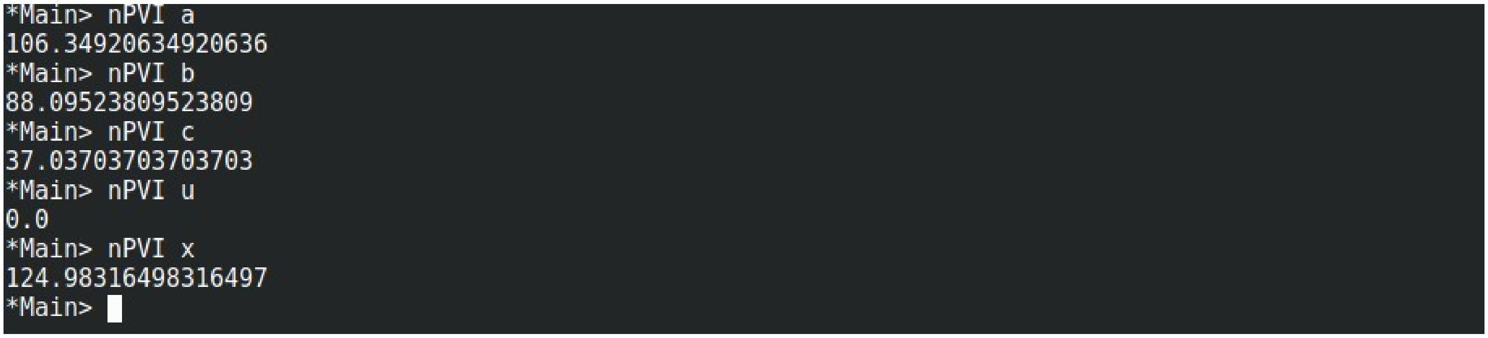
```
*Main> pairwiseTermPairs a
[(-3,3.5),(4,3.0),(-2,2.0)]
*Main> pairwiseTermPairs b
[(-2,2.0),(-3,4.5),(4,4.0),(-3,3.5)]
*Main> pairwiseTermPairs c
[(0,4.0),(2,3.0),(1,1.5),(0,1.0),(-1,1.5),(0,2.0),(-2,3.0),(0,4.0),(-4,6.0)]
*Main> pairwiseTermPairs u
[(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0),(0,2.0)]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

Task 7h - The pairwiseTerms function



```
*Main> pairwiseTerms a
[0.8571428571428571,1.3333333333333333,1.0]
*Main> pairwiseTerms b
[1.0,0.6666666666666666,1.0,0.8571428571428571]
*Main> pairwiseTerms c
[0.0,0.6666666666666666,0.6666666666666666,0.0,0.6666666666666666,0.0,0.6666666666666666,0.0,0.66666
66666666666]
*Main> pairwiseTerms u
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
*Main> pairwiseTerms x
[1.6,1.2727272727272727,1.2,0.9090909090909091,0.8,1.1111111111111112,1.2,1.5555555555555556,1.6]
*Main> 
```

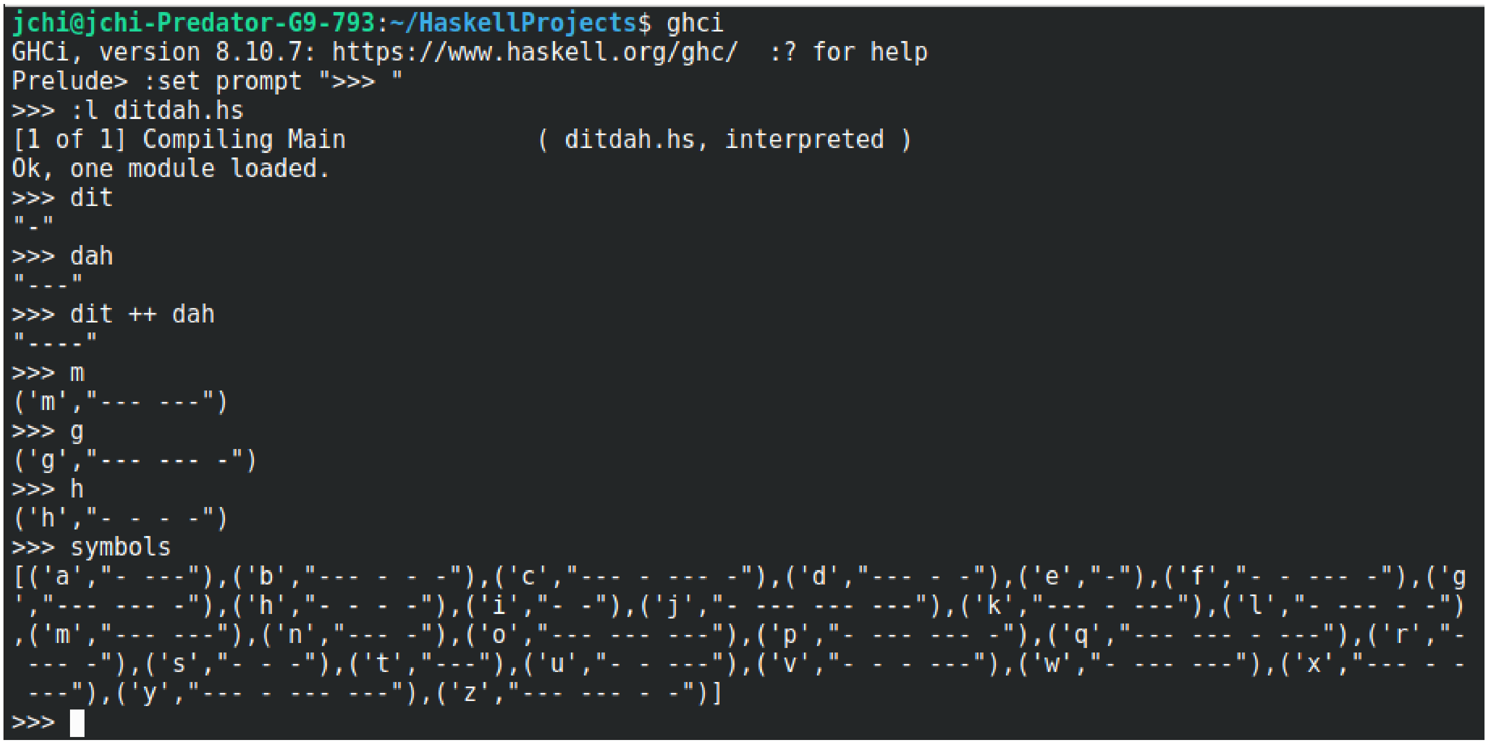
Task 7i - The nPVI function



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

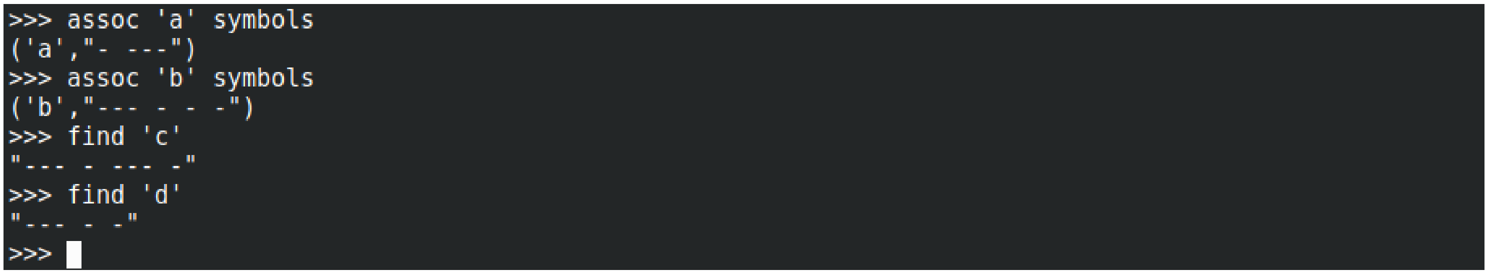
Task 8 - Historic Code: The Dit Dah Code

Subtask 8a



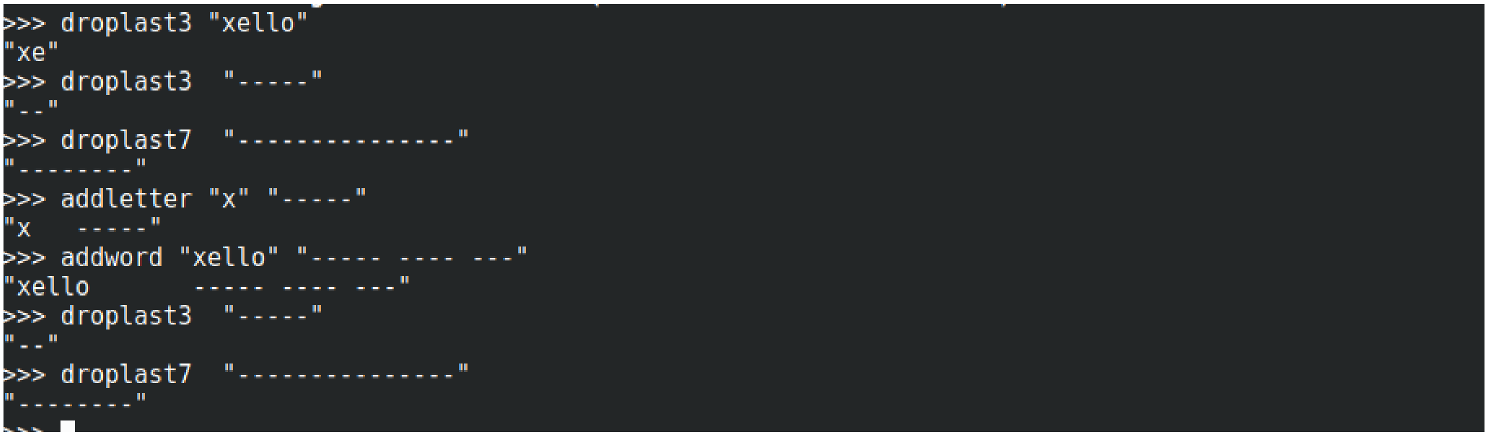
```
jchi@jchi-Predator-G9-793:~/HaskellProjects$ ghci
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
Prelude> :set prompt ">>> "
>>> :l ditdah.hs
[1 of 1] Compiling Main             ( ditdah.hs, interpreted )
Ok, one module loaded.
>>> dit
"-"
>>> dah
"---"
>>> dit ++ dah
"----"
>>> 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



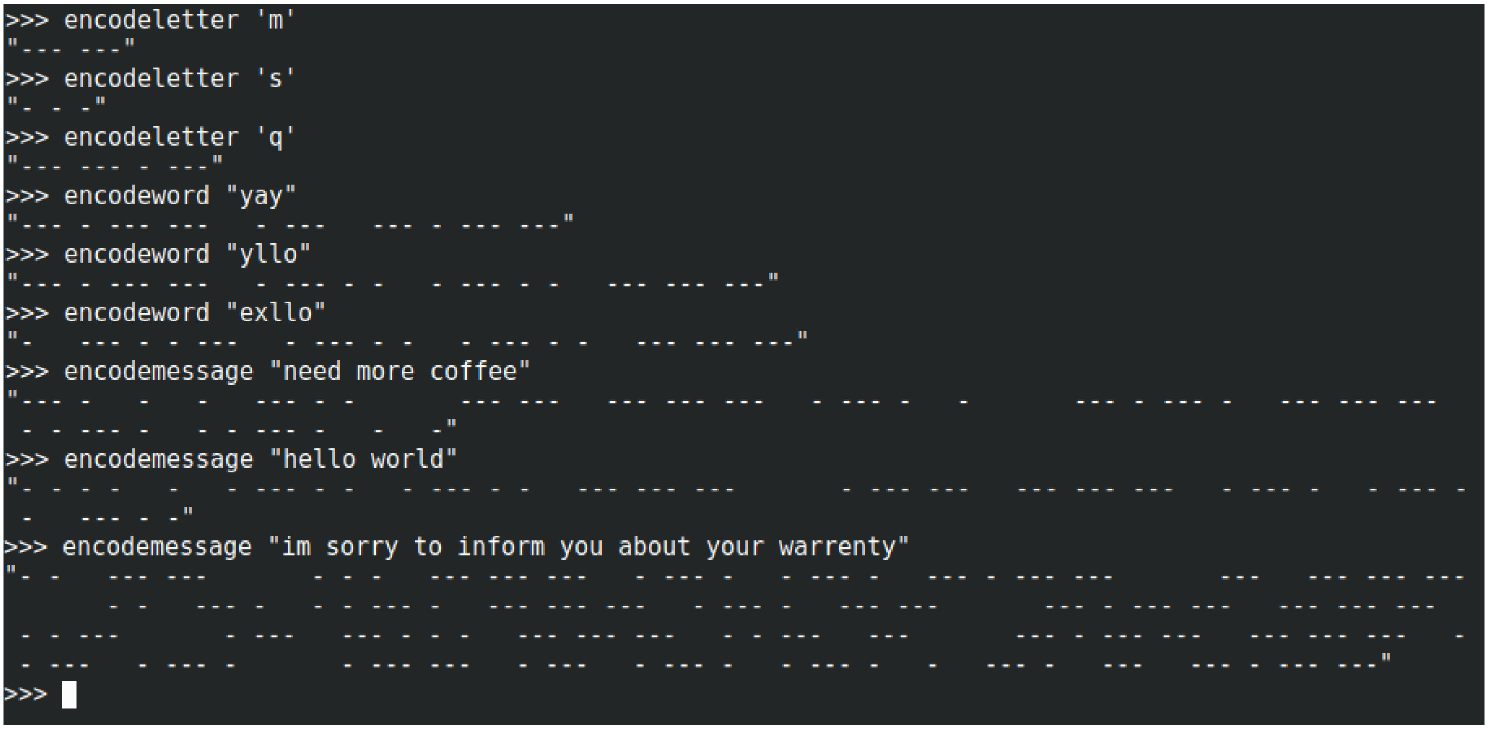
```
>>> assoc 'a' symbols
('a',"- ---")
>>> assoc 'b' symbols
('b',"--- - - -")
>>> find 'c'
"--- - --- -"
>>> find 'd'
"--- - -"
>>> 
```

Subtask 8c



```
>>> droplast3 "xello"
"xe"
>>> droplast3  "-----"
"--"
>>> droplast7  "---------------"
"--------"
>>> addletter "x" "-----"
"x   -----"
>>> addword "xello" "----- ---- ---"
"xello       ----- ---- ---"
>>> droplast3  "-----"
"--"
>>> droplast7  "---------------"
"--------"
>>> 
```

Subtask 8d



```
>>> encodeletter 'm'
"--- ---"
>>> encodeletter 's'
"- - -"
>>> encodeletter 'q'
"--- --- - ---"
>>> encodeword "yay"
"--- - --- ---   - ---   --- - --- ---"
>>> encodeword "yllo"
"--- - --- ---   - --- - -   - --- - -   --- --- ---"
>>> encodeword "exllo"
"-   --- - - ---   - --- - -   - --- - -   --- --- ---"
>>> encodemessage "need more coffee"
"--- -   -   -   --- - -       --- ---   --- --- ---   - --- -   -       --- - --- -   --- --- ---
 - - --- -   - - --- -   -   -"
>>> encodemessage "hello world"
"- - - -   -   - --- - -   - --- - -   --- --- ---       - --- ---   --- --- ---   - --- -   - --- -
 -   --- - -"
>>> encodemessage "im sorry to inform you about your warrenty"
"- -   --- ---       - - -   --- --- ---   - --- -   - --- -   --- - --- ---       ---   --- --- ---
       - -   --- -   - - --- -   --- --- ---   - --- -   --- ---       --- - --- ---   --- --- ---
 - - ---       - ---   --- - - -   --- --- ---   - - ---   ---       --- - --- ---   --- --- ---   -
 - ---   - --- -       - --- ---   - ---   - --- -   - --- -   -   --- -   ---   --- - --- ---"
>>> 
```