Haskell Programming Assignment: Various Computations

Abstract: This assignment focuses on functions, recursive list processing, list comprehensions, and high order functions in Haskell. This assignment has 8 tasks. Some tasks are broken down into subtasks

Interaction: Task 1 Mimicking the Demo

```
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'

True

>>> ( \x -> x /= ' ' ) ' '

False

>>> filter ( \x -> x /= ' ' ) "Is the Haskell fun yet?"

"IstheHaskellfunyet?"

>>> :quite

unknown command ':quite'

use :? for help.

>>> :quit

Leaving GHCi.
```

Interaction: Task 2 Numeric Functions

Demo:

```
>>> :load ha

[1 of 1] Compiling Main           ( ha.hs, interpreted )


ha.hs:29:1: warning: [-Wtabs]

    Tab character found here, and in 16 further locations.

    Please use spaces instead.

   |

29 |                where sumSideArea = 6 * squareArea side

   | ^^^^^^^^

Ok, one module loaded.
```

```
>>> squareArea 10

100

>>> squareArea 12

144

>>> 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

0.0

>>> paintedCube2 2

0.0

>>> paintedCube2 3
```

```
12.0
>>> map paintedCube2 [1..10]
[0.0,0.0,12.0,24.0,36.0,48.0,60.0,72.0,84.0,96.0]
>>> :quit
Leaving GHCi.
```

Code:

```
--------------------------------------
--------------------------------------
--- Task 2 Define 5 functions squareArea, circleArea, blueAreaOfCube,
paintedCube1, paintedCube2



--------------------------------------
--------------------------------------
--- squareArea

squareArea side = side * side

--------------------------------------
--------------------------------------
--- circleArea

circleArea radius = pi * ( radius ^ 2 )

--------------------------------------
--------------------------------------
--- blueAreaOfCube

blueAreaOfCube side = sumSideArea - sumDotArea
            where sumSideArea = 6 * squareArea side
```

```
                    sumDotArea = 6 * circleArea ( side / 4 )



-------------------------------------

-------------------------------------

--- paintedCube1


paintedCube1 1 = 0

paintedCube1 2 = 0

paintedCube1 n = total

          where total = ( ( ( n ^ 2 ) * 6 ) + noSide ) - ( twoSide + 
threeSide + noSide )

              twoSide = ( ( ( n - 2 ) * 4 ) * 6 )

              threeSide = ( 4 * 6 )

              noSide = ( n - 2 ) ^ 3



-------------------------------------

-------------------------------------

--- paintedCube2


paintedCube2 1 = 0

paintedCube2 2 = 0

paintedCube2 n = total / 2

          where total = ( ( ( n ^ 2 ) * 6 ) + noSide ) - ( oneSide + 
threeSide + noSide )

              oneSide = paintedCube1 n

              threeSide = ( 4 * 6 )

              noSide = ( n - 2 ) ^ 3
```

Interaction: Task 4 Recursive List Processors

1.

Demo:

```
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"
```

Code:

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

-------------------------------------

--- Task 4 Recursive List Processors

--- List2set, isPalindrome


list2set [] = []

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

Interaction: Task 5 List Comprehensions

1.

Demo:

```
>>> count 'e' "need more coffee"

5

>>> count 4 [1,2,3,2,3,4,3,4,5,4,5,6]

3
```

Code:

```
count e l = length [ x | x <- l , x == e ]
```

2.

Demo:

```
>>> 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)]
```

Code:

```
freqTable list = [ (e, count e list ) | e <- list2set list ]
```

Interaction: Task 6 Higher Order Functions

1.

Demo:

```
>>> tgl 5
15
>>> tgl 10
55
```

Code:

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

2.

Demo:

```
>>> 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]
```

Code:

```
triangleSequence n = map tgl [1..n]
```

3.

Demo:

```
>>> vowelCount "cat"
1
>>> vowelCount "mouse"
3
```

Code:

```
vowelCount word = length ( filter ( \l -> l `elem` "aeiou" ) word )
```

4.

Demo:

```
>>> 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]
```

Code:

```
lcsim f p xs = map f $ filter p xs
```

Interaction: Task 7 An Interesting Statistic: nPVI

Interaction: Task 8Historic Code: The Dit Dah Code