19CSE313 - Principles of Programming Languages

Lab – 7

SHOAIB AKHTAR
AM.EN.U4CSE20163

1) Write a Haskell function **squareAll** :: [Int] → [Int] which takes a list of integers and produces a list of the squares of those integers.

squareList :: [Int] -> [Int]

squareList xs = map (\x -> x^2) xs

```
Main> squareList [2,3,4,5,6,7]
[4,9,16,25,36,49]
Main>
```

2) Write a Haskell function to **capitalize all the letters** in the list of characters using map function.

import Data.Char

capitalizeList :: [Char] -> [Char]

capitalizeList xs = map toUpper xs

```
Main> capitalizeList "abcdefgh"
"ABCDEFGH"
Main> capitalizeList "hello , how was the day ?"
"HELLO , HOW WAS THE DAY ?"
Main>
```

3) Write a Haskell function **nestReverse** which takes a list of strings as its argument and reverses each element of the list and then reverses the resulting list.

nestReverse :: [String] -> [String]

nestReverse = reverse . map reverse

```
Main> nestReverse ["foo", "bar", "baz"]
["zab","rab","oof"]
Main> nestReverse ["hii", "hello", "when"]
["nehw","olleh","iih"]
Main>
```

4) Write a Haskell function **inFront**:: a → a → a which takes an object and a list of lists and sticks the object at the front of every component list.

inFront :: a -> [[a]] -> [[a]]

inFront x = map (x:)

```
Main> inFront 'a' ["foo", "bar", "baz"]
["afoo","abar","abaz"]
Main> inFront 1 [[2, 3], [4, 5], [6, 7, 8]]
[[1,2,3],[1,4,5],[1,6,7,8]]
Main>
```

5) Write a Haskell function **strLengths** which takes a list of strings as its argument and returns the list of their lengths.

strLengths :: [String] -> [Int]

strLengths = map length

```
Main> strLengths ["foo", "bar", "baz"]
[3,3,3]
Main> strLengths ["hello", "world", "haskell"]
[5,5,7]
Main>
```

6) Write a Haskell function **parityList** :: [String] → [Int] which takes a list of strings and returns a list of the integers 0 and 1 such that 0 is the n^{th} element of the list returned, if the n^{th} string of the argument contains an even number of characters and 1 is the nth element of the list returned, if the n^{th} string contains an odd number of characters.

parityList :: [String] -> [Int]

parityList = map (\s -> if even (length s) then 0 else 1)

```
Main> parityList ["foo", "bar", "baz"]
[1,1,1]
Main> parityList ["hello", "world", "!", "Haskell"]
[1,1,1,1]
Main> parityList ["hi","Haskell"]
[0,1]
Main>
```

7) Using the higher-order function map define a function **sumsq** which takes an integer n as its argument and returns the sum of the squares of the first n integers.

sumsq :: Int -> Int

sumsq n = sum (map (^2) [1..n])

```
Main> sumsq 5
55
Main> sumsq 30
9455
Main> sumsq 12
650
Main>
```

8) Write a Haskell function **noCapitals** which removes all the capital letters from a string.

import Data.Char (isUpper)

noCapitals :: String -> String

noCapitals = filter (\c -> not (isUpper c))

```
Main> noCapitals "Hello, World!"
"ello, orld!"
Main> noCapitals "hAsKeLl"
"hsel"
Main>
```

9) Write a Haskell function **removeVowels** which takes a list of strings as its argument and removes every occurrence of a vowel from each element.

removeVowels :: [String] -> [String]

removeVowels = map (filter (`notElem` "aeiouAEIOU"))

```
Main> removeVowels ["hello", "world", "Haskell"]
["hll","wrld","Hskll"]
Main> removeVowels ["original"]
["rgnl"]
Main>
```

10) Write a Haskell function **noVowel**(without vowels) which removes every occurrence of a vowel from a list of characters.

noVowel :: [Char] -> [Char]

noVowel = filter (`notElem` "aeiouAEIOU")

```
Main> noVowel "Hello, Haskell!"
"Hll, Hskll!"
Main> noVowel "My name is Harry"
"My nm s Hrry"
Main>
```

11) Write a Haskell function **rotateABC** that changes a's to b's, b's to c's and c's to a's in a String. Only lowercase are affected.

rotateABC :: String -> String

rotateABC = map rotateChar

where

rotateChar 'a' = 'b'

rotateChar 'b' = 'c'

rotateChar 'c' = 'a'

rotateChar c = c

```
Main> rotateABC "abcabc"
"bcabca"
Main> rotateABC "bacabbac"
"cbabccba"
Main>
```