LAB SHEET - 7

NAME : J.SAMPAT SRIVATSAV

ROLL NUMBER : AM.EN.U4CSE19125

BATCH : C.S.E – B

1. Write a Haskell function **squareAll** :: [Int] → [Int] which takes a list of integers and produces a list of the squares of those integers. For example, squareAll [6, 1, (-3)] = [36, 1, 9].

```
squareAll :: [Int] -> [Int]
squareAll xs = map (\x -> x^2) xs
```

```
adhinene@sampat: ~/ppl/lab7
*Main> squareAll [6,1,(-3)]
[36,1,9]
*Main> squareAll [6,1,3]
[36,1,9]
*Main> squareAll [1..100]
[1,4,9,16,25,36,49,64,81,100,121,144,169,196,225,256,289,324,361,400,441,484,529
,576,625,676,729,784,841,900,961,1024,1089,1156,1225,1296,1369,1444,1521,1600,16
81,1764,1849,1936,2025,2116,2209,2304,2401,2500,2601,2704,2809,2916,3025,3136,32
49,3364,3481,3600,3721,3844,3969,4096,4225,4356,4489,4624,4761,4900,5041,5184,53
29,5476,5625,5776,5929,6084,6241,6400,6561,6724,6889,7056,7225,7396,7569,7744,79
21,8100,8281,8464,8649,8836,9025,9216,9409,9604,9801,10000]
*Main>
```

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

```
import Data.Char
capitalize :: String -> String
capitalize s = map (\x -> toUpper x) s
```

```
adhinene@sampat: ~/ppl/lab7
*Main> capitalize "sampat"
"SAMPAT"
*Main> capitalize "iygweyd"
"IYGWEYD"
*Main> capitalize "HJGKUGWhguguUGYUF"
"HJGKUGWHGUGUUGYUF"
*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. Thus, nestReverse ["in", "the", "end"] = ["dne", "eht", "ni"].

```haskell
nestReverse :: [String] -> [String]
nestReverse xs = reverse [reverse x | x <- xs]
```

```
adhinene@sampat: ~/ppl/lab7
*Main> nestReverse ["in","the","end"]
["dne","eht","ni"]
*Main> nestReverse ["Sampat","Sri","Vatsav"]
["vastaV","irS","tapmaS"]
*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. For example, **inFront** 7 [[1,2], [], [3]] = [[7,1,2], [7], [7,3]].

```haskell
inFront :: a -> [[a]] -> [[a]]
inFront x xxs = [x:xs | xs <- xxs]
```

```
adhinene@sampat: ~/ppl/lab7
*Main> inFront 7 [[1,2],[],[3]]
[[7,1,2],[7],[7,3]]
*Main> inFront 1 [[2..10],[],[2,5..10]]
[[1,2,3,4,5,6,7,8,9,10],[1],[1,2,5,8]]
*Main>
```

5. Write a Haskell function **strLengths** which takes a list of strings as its argument and returns the list of their lengths. For example, strLengths ["the", "end", "is", "nigh"] = [3, 3, 2, 4].

```haskell
strLengths :: [String] -> [Int]
strLengths ss = map (\x -> length x) [t | t <- ss]
```

```
adhinene@sampat: ~/ppl/lab7
*Main> strLengths ["the","end","is","nigh"]
[3,3,2,4]
*Main> strLengths ["Jakkinapalli","Sampat","Sri","Vatsav"]
[12,6,3,6]
*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 nth element of the list returned, if the nth string of the argument contains an even number of characters and 1 is the nth element of the list returned, if the nth string contains an odd number of characters. For example, **parityList ["one", "two", "three", "four"] = [1, 1, 1, 0].**

```haskell
parityList :: [String] -> [Int]
parityList ss = map f [s | s <- ss]
                where f s=if(length s `mod` 2 == 0) then 0
                          else 1
```

```
adhinene@sampat: ~/ppl/lab7
*Main> parityList ["one","two","three","four"]
[1,1,1,0]
*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. That is to say, **sumsq n = 1^2 +2^2 +3^2+...+n^2**.

```haskell
sumsq :: Int -> Int
sumsq x = sum (map (\x -> x^2) [1..x])
```

```
adhinene@sampat: ~/ppl/lab7
Prelude> :l q7.hs
[1 of 1] Compiling Main             ( q7.hs, interpreted )
Ok, one module loaded.
*Main> sumsq 10
385
*Main> sumsq 5
55
*Main> sumsq 2
5
*Main> sumsq 3
14
*Main> sumsq 10000
333383335000
*Main>
```

8. Write a Haskell function **noCapitals** which removes all the capital letters from a string. Thus, noCapitals "Amrit Kiran" = "mrit iran".

```haskell
noCapitals :: String -> String
noCapitals ss = [s | s <- ss , not(s `elem` ['A'..'Z'])]
```

```
adhinene@sampat: ~/ppl/lab7
*Main> noCapitals "Amrita Kiran"
"mrita iran"
*Main> noCapitals "Sampat Srivatsav"
"ampat rivatsav"
*Main> noCapitals "sampat srivatsav"
"sampat srivatsav"
*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. For example, removeVowels ["the", "end", "is", "nigh"] = ["th", "nd", "s", "ngh"].

```haskell
removeVowels :: [String] -> [String]
removeVowels xss = map f [xs | xs <- xss]

                   where
                        f "" = ""
                        f (x:xs) = if(x `elem` ['a','e','i','o','u']) then f xs
                                   else x:f xs
```

```
adhinene@sampat: ~/ppl/lab7
*Main> removeVowels ["the","end","is","nigh"]
["th","nd","s","ngh"]
*Main>
```

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

```haskell
noVowel :: [Char] -> [Char]
noVowel xs = [x | x <- xs , not (x `elem` ['a','e','i','o','u'])]
```

```
adhinene@sampat: ~/ppl/lab7
Prelude> :l q10.hs
[1 of 1] Compiling Main             ( q10.hs, interpreted )
Ok, one module loaded.
*Main> noVowel ['a','e','i','o','u']
""
*Main> noVowel ['s','a','m','p','a','t']
"smpt"
*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.

```haskell
rotateABC :: String -> String
rotateABC "" = ""
rotateABC (x:xs) = if(x=='a') then 'b':rotateABC xs
                   else if(x=='b') then 'c':rotateABC xs
                   else if(x=='c') then 'a':rotateABC xs
                   else x:rotateABC xs
```

adhinene@sampat: ~/ppl/lab7

```
*Main> rotateABC "sampat"
"sbmpbt"
*Main> rotateABC "abbbyqgwdbgxxccc123"
"bcccyqgwdcgxxaaa123"
*Main>
```