19CSE313

Principles of Programming Languages

Lab 7

S Abhishek

AM.EN.U4CSE19147

1 - Take a list of integers and produces a list of the squares of those integers.

```
squareAll :: [Int] -> [Int]

squareAll x = map (^2) x
```

```
*Main> squareAll [6,1,(-3)]
[36,1,9]
*Main> squareAll [2,2,2]
[4,4,4]
```

2 - Capitalize all the letters in the list of characters

```
capL :: [Char] -> [Char]

capL x  = map toUpper x
```

```
*Main> capL ['a','b','c']
"ABC"
*Main> capL ['y','z']
"YZ"
```

3 - Take a list of strings as its argument and reverses each element of the list and then reverses the resulting list.

```
nestReverse :: [String] -> [String]

nestReverse x = map reverse (reverse x)
```

```
*Main> nestReverse ["Abhi", "Hello"]
["olleH","ihbA"]
*Main> nestReverse ["in", "the", "end"]
["dne","eht","ni"]
```

4 - Take an object and a list of lists and sticks the object at the front of every component list.

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

inFront x y = map ( x : ) y
```

```
*Main> inFront 7 [[1,2],[],[3]]
[[7,1,2],[7],[7,3]]
*Main> inFront 0 [[1,2],[1],[1,2,3,4,5]]
[[0,1,2],[0,1],[0,1,2,3,4,5]]
```

5 - Take a list of strings as its argument and returns the list of their lengths.

```
strLengths :: [String] -> [Int]

strLengths x = map length x
```

```
*Main> strLengths ["the", "end", "is", "nigh"]
[3,3,2,4]
*Main> strLengths ["Abhi", ""]
[4,0]
```

6 - Take a list of strings and returns a list of the integers 0 and 1 such that 0 is the nth element of the value if the nth string of the argument contains an even number of characters and 1 is the nth element of the value if the nth string contains an odd number of characters.

```
parityList :: [String] -> [Int]

parityList x = map check x where

check x | even (length x) = 0

| otherwise = 1
```

```
*Main> parityList ["one", "two", "three", "four"]
[1,1,1,0]
*Main> parityList ["Abhi", "Bharath", "Chinnu"]
[0,1,0]
```

7 - Take an integer n as its argument and returns the sum of the squares of the first n integers.

```
sumsq :: Int -> Int

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

```
*Main> sumsq 2
5
*Main> sumsq 4
30
*Main> sumsq 5
55
```

8 - Remove all the capital letters from a string

```
noCapitals :: String -> String

noCapitals x = [ i | i <- x, i `notElem` ['A' .. 'Z'] ]
```

```
*Main> noCapitals "Amrit Kiran"
"mrit iran"
*Main> noCapitals "Abhi"
"bhi"
*Main> noCapitals "a3x3k"
"a3x3k"
```

```
remUpper :: String -> String

remUpper = filter isLower
```

```
*Main> remUpper "Abhi"
"bhi"
*Main> remUpper "Amrit Kiran"
"mritiran"
*Main> remUpper "A"
""
*Main> remUpper ""
""
```

9 - Take a list of strings as its argument and removes every occurrence of a vowel from each element.

```
checkVowel :: String -> String

checkVowel x = [ i | i <- x, i `notElem` ['a','e','i','o','u','A','E','I','O','U'] ]

removeVowels :: [String] -> [String]

removeVowels x = map checkVowel x
```

```
*Main> removeVowels ["the", "end", "is", "nigh"]
["th","nd","s","ngh"]
*Main> removeVowels ["a", "e", "i", "b"]
["","","","b"]
```

```
cVow :: String -> String

cVow x = filter (\x -> not (x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u')) x

remVowels :: [String] -> [String]

remVowels = map cVow
```

```
*Main> remVowels ["the", "end", "is", "nigh"]
["th","nd","s","ngh"]
*Main> remVowels ["Hello", "end", "eiu", "a"]
["Hll","nd","",""]
```

10 - Remove every occurrence of a vowel from a list of characters

```
remVow :: [Char] -> [Char]

remVow x = filter (\x -> not (x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u')) x
```

```
*Main> remVow ['a', 'b', 'c', 'e', 'r', 'i', 'p']
"bcrp"
*Main> remVow ['a', 'e', 'i']
""
*Main> remVow ['a', 'e']
""
*Main> remVow ['b']
"b"
*Main> remVow []
""
```

11 - Changes lower case a’s to b’s, b’s to c’s and c’s to a’s in a String.

```
rotabc = map f where
 f 'a' = 'b'
 f 'b' = 'c'
 f 'c' = 'a'
 f c = c
```

```
*Main> rotabc "abhi"
"bchi"
*Main> rotabc "abcd"
"bcad"
*Main> rotabc "ABCd"
"ABCd"
*Main> rotabc "Dog"
"Dog"
```