**20CYS312 - PPL - Lab Exercise 3**

**Name:Prasanna N R Date:20-12-2024**  
**Reg.No:CH.EN.U4CYS22039**

**Objective:**  
The goal is to write functions to perform arithmetic operations, Tuple Operations, list manipulations, basic function definitions.

**1. List and Tuples**

**a. Swap two elements in a Tuple**

**Task**: Implement a function swapTuple that takes a tuple (a, b) and swaps its elements, i.e., returns the tuple (b, a).

**Solution**:

swap (x,y) = (y,x)

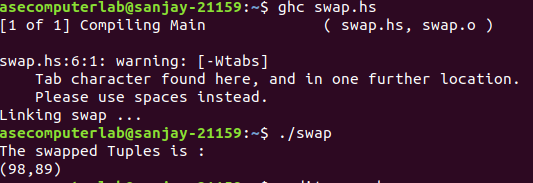
main :: IO()

main = do

putStrLn "The swapped Tuples is :"

print(swap (89,98))

**Explanation**: The function swap is used to swap two elements in a tuple and Prints the result

**Output:**

**b. Multiply Elements in a Lists**

**Task**: Write a function multiplyElements that takes a list of numbers and a multiplier n, and returns a new list where each element is multiplied by n. Use a list comprehension for this task.

**Solution**:

multi :: [Int] ->Int -> [Int]

multi nums n = [x\*n | x <- nums]

main :: IO()

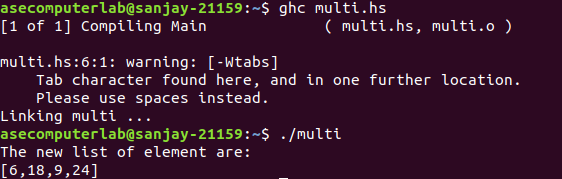
main= do

putStrLn "The new list of element are:"

print(multi [2,6,3,8] 3)

**Explanation**: The function Multi is used to multiple a all the elements in the lists with a number n given by the user andreturns the new lists.

**Output:**



**c. Filter Even Numbers**

**Task**: Write a function filterEven that filters out all even numbers from a list of integers using the filter function.

**Solution**:

filterEven :: [Int] -> [Int]

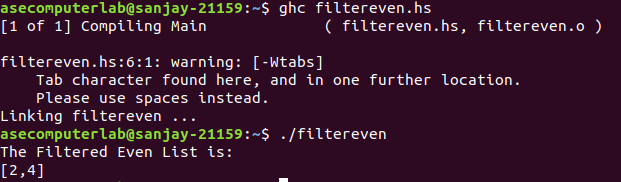
filterEven xs = filter even xs

main :: IO ()

main = do

print (filterEven [1, 2, 3, 4])

**Explanation**: The function filterEven uses the built-in filter function, which applies the even function to each element of the list and keeps only those that return True.

**Output:**

**d. List Zip with**

**Task:**Implement a function listZipWiththat behaves similarly to zipWithin Haskell. It should take a function and two lists, and return a list by applying the function to corresponding elements from both lists.

**Solution:**

listZipWith :: (a -> b -> c) -> [a] -> [b] -> [c]

listZipWith f [] \_ = []

listZipWith f \_ [] = []

listZipWith f (x:xs) (y:ys) = f x y : listZipWith f xs ys

main :: IO ()

main = do

let nums1 = [1, 2, 3, 4, 5]

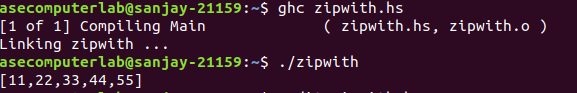
let nums2 = [10, 20, 30, 40, 50]

let result = listZipWith (+) nums1 nums2

print result

**Explanation**: The function Zipwith used to add the elements in a two lists and combine it and prints the results.

**Output:**



**e. Reverse a List**

**Task**: Define a function reverseList that takes a list and returns a new list with the elements in reverse order.

**Solution**:

reverseList :: [a] -> [a]

reverseList = reverse

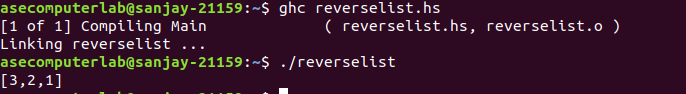
main :: IO ()

main = do

print (reverseList [1, 2, 3])

**Explanation**: The function reverlist will uses a in-bulit function reverse and reverses the list and returns the new list

**Output:**



**f. Creating a student Database**

**Task**: You are tasked with developing a program to manage and analyze student records. Each student is represented as a tuple (String, Int, [Int]), where the first element is the student’s name (a string), the second is their roll number (an integer), and the third is a list of integers representing their marks in various subjects. Write a recursive function averageMarks to calculate the average of a student's marks. Display all student names and their average marks.

**Solution**:

import Data.List (sortBy)

type Student = (String, Int, [Int])

averageMarks :: Student -> Double

averageMarks (name, rollNo, marks) = fromIntegral (sum marks) / fromIntegral (length marks)

displayStudents :: [Student] -> IO ()

displayStudents [] = return ()

displayStudents ((name, rollNo, marks):students) = do

putStrLn $ name ++ ": " ++ show (averageMarks (name, rollNo, marks))

displayStudents students

students :: [Student]

students = [

("Prasanna", 1, [95, 92, 98, 90]),

("Harishkar", 3, [90, 88, 92, 85]),

("Pranish", 2, [75, 80, 85, 82]),

("David Billa", 4, [80, 75, 82, 78])]

main :: IO ()

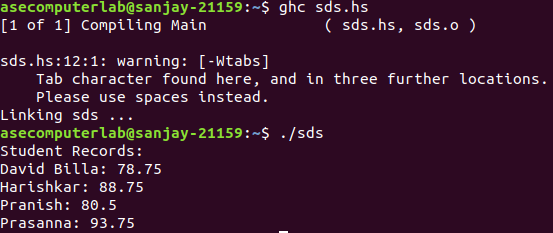
main = do

putStrLn "Student Records:"

displayStudents (sortBy (\(a,\_,\_) (b,\_,\_) -> compare a b) students)

**Explanation**: The function students used to store the data of students that is Name, Marks,Age and Returns the Average of Each Students

**Output:**



**Conclusion:**

This lab exercise helped understand fundamental concepts of Haskell, including list and Tuple operations,basic Student Data System.