Exercise 4

Collections in Scala

- Last Week Recap [**Higher Order Functions** (*a function that takes another function as an argument or returns a function as its result*) & **Lists** (*represents a finite number of ordered values, where the same value may occur more than once*)]
- Collections(*A collection is a single object representing a group of objects and they should be used when we want to associate some significant **behaviour** with the data in the collection*)
- Sequence (*A sequence is a collection in which there is a specific order to the elements it holds*)
- Set (A *set does not allow a duplicate of an element*)
- Map (*A map is a collection that maintains relationships between keys and values*)

Classwork

In this lab, you will be exploring the fundamentals of different collections in Scala. You will learn about the syntax for different collection behaviors (i.e. Sequences, Sets, Maps) and practice how to use Higher Order Functions (HOFs) and different collections together with built-in behaviors.

Homework

NOTE: You should use immutable lists and avoid using any mutable variables or loops.

Task 0: Sequences, Sets & Maps built-in Functions Practice

- Create a list of integer **nums** containing the numbers 1 to 10.
- Use the **map** method to create a new list **squares** that contains the squares of each number in **nums**.
- Use the **filter** method to create a new list **evenSquares** that contains only the even squares from the **squares** list.
- Create a set **evenSet** that contains the even numbers from the **nums** list.
- Use the **foldLeft** method to calculate the sum of all the elements in the **evenSet**.
- Create a map **squaresMap** that maps each number in nums to its square.
- Use the **getOrElse** method on the **squaresMap** to get the square of the number 11. If the number is not present in the map, return 0.

Reference: Common Collection Behaviors, Page No. 259 & 260 [Chapter 25, ***A Beginner's Guide to Scala, Object Orientation and Functional Programming*****, Hunt, 2018]**

Task 1: Write a function **removeDuplicates** that removes duplicates from a **sequence** of strings. This function should take a sequence of strings **seq** as input and uses the **foldLeft** method to accumulate a new sequence **newSeq** that contains only the unique elements from the input sequence.
Example Input: “hello", "world", "hello", "scala", "world"

Example Output: “hello", "world", "scala"

Task 2: Write a function **findCommonElements** that finds the common elements in two different sets. This function should takes two sets **set1** and **set2** of elements of type **A** as input and returns a new set that contains the elements that are common to both input sets.
Example Input: (1, 2, 3, 4, 5) & (3, 4, 5, 6, 7)
Example Output: (3, 4, 5)

Task 3: Write a function **removeKeys** that removes elements from an **immutable** map. The function should take a map and a list of keys to remove as input and return a new map that does not contain the specified keys.

Task 4: Write a Scala program that reads a text file and counts the occurrence of each word in the file. You should use an **immutable** map to store the word frequencies.

Example Input:
This is a test file.
It contains several lines.
Each line has some words.
Some words are repeated.
Others appear only once.
Example Output:
This 1
is 1
a 1
test 1
file. 1
It 1
contains 1
several 1
lines. 1
Each 1
line 1
has 1
some 2
words. 1
Some 1
words 1
are 1
repeated. 1
Others 1
appear 1
only 1
once. 1

Deliverable

Deliverable: Submit a single scala code file with "*.scala*" extension, and write your Name and Student ID in the code as a comment. The whole deliverable must be well commented and supported with descriptions where required.
Deadline: 14.04.2023 12:00 am [Before Next Friday Session]
Submission: (session respective) Return box on Moodle.
Estimated workload: <= 2 hours
Warning: This is individual work. Strict actions will be taken for plagiarism!

Deliverables for Exercise 4:

1. Implementation of the homework part.

Note:
Make sure to follow the Scala style guide and use appropriate naming conventions for variables, functions, and classes. Your code should be well-organized, well-documented, and easy to read.