List Processing I

To begin:

· In Eclipse create a Scala project called ListLab.

· Add a worksheet to this project called session

· Add a Scala interpreter to this project

In the session worksheet define and test the following functions.

For problems 1, 2, 6, 7, & 8 implement four versions:

· Iterative version

· Recursive version

· Tail-recursive version (this should be different from the previous version)

· map-filter-reduce version

All of your implementations should be as generic as possible.

To end:

· Export your worksheet to your file system and send it to the grader by the deadline.

1. Problem

Write a function that computes the sum of cubes of all odd numbers occurring in a list of integers.

2. Problem

Write a function that computes the sum of numbers in a list of lists of numbers:

```
sumOfSums(List(List 1, 2, 3), List(4, 5, 6)) = 21
```

3. Problem

Write a function that returns the depth of a list of nested lists:

```
depth(List(List(List 1, 2, List(3)))) = 4
```

4. Problem

Write a function that computes the average of a list of doubles

5. Problem

Write a function that returns the largest element of a list of comparables.

6. Problem

Write a function that returns the number of elements in a list that satisfy a given predicate. (The predicate is a parameter of type T=>Boolean.)

7. Problem

Write a function that returns true if all elements in a list satisfy a given predicate.

8. Problem

Write a function that returns true if any element in a list satisfies a given predicate.

9. Problem

Write a function that reverses the elements in a list.

10. Problem

Write a function that returns true if a given list of integers is sorted (in ascending order).

11. Problem: My Map

If the List.map function didn't exist, how would you define it?

12. Problem: Take, Drop and Zip

If take, drop, and zip didn't exist for lists, how would you define them?

13. Problem: Streams

A stream is like a list except that it is constructed using the lazy version of cons:

```
scala>  val s1 = 1 #:: 2 #:: 3 #:: Stream.Empty
s1: scala.collection.immutable.Stream[Int] = Stream(1, ?)

scala> s1.head
res0: Int = 1

scala> s1.tail.head
res1: Int = 2
```

Create the following streams

- An infinitely long stream of 1's
- The stream of all non-negative integers
- The stream of all non-negative even integers
- The stream of all squares of integers