Exercise 3

This exercise session consists of Scala programming exercises.

Higher Order Functions Basics

Implement the methods described in the file: higherOrderFunc.scala.

Exercises From Last Week OPTIONAL

Revisit the exercises from last week and think about how you could use higher-order functions to solve the problems.

Hint: map, flatmap, pack, span and foldLeft are relevant.

Binary Search Trees (Dictionaries)

Background Information

A binary tree is either empty or it is composed of a root element and two successors, which are binary trees themselves.

A basic initial implementation of a binary tree is available in this project. Go to the binarytree.scala file and look at the following description of the implementation:

An End is equivalent to an empty tree. A Branch has a value, and two descendant trees.

A tree with only a root node would be Node(4) and an empty tree would be End.

Implementation: addNode

Write a function to add an element to a binary search tree.

```
scala> End.addNode(2)
res0: Node[Int] = T(2 . .)

scala> res0.addNode(3)
res1: Node[Int] = T(2 . T(3 . .))

scala> res1.addNode(0)
res2: Node[Int] = T(2 T(0 . .) T(3 . .))
```

Hint: The abstract definition of addNode in Tree should be:

```
def addNode(x:Int,tree:BinaryTree):BinaryTree
```

Implementation: isBinarySearchTree

Write a function to check if the given BinaryTree is a binary search tree. The function should have the following definition:

```
def isBinarySearchTree(bt:BinaryTree):Boolean
```

The exercise is adapted from P57 at http://aperiodic.net/phil/scala/s-99/.

Gray Code

Create a new file and implement P49 from http://aperiodic.net/phil/scala/s-99/ as described in the following.

An n-bit Gray code is a sequence of n-bit strings constructed according to certain rules. For example, n = 1: C(1) = ("0", "1"). n = 2: C(2) = ("00", "01", "11", "10"). n = 3: C(3) = ("000", "001", "011", "010", "110", "111", "101", "100"). Find out the construction rules and write a function to generate Gray codes.

For instance:

```
gray(3)
res0 List[String] = List(000, 001, 011, 010, 110, 111, 101, 100)
```