Recitation 4

Higher-Order Functions

Scala Built-Ins:

- `map [A] (f : (A) => B) : List[B]`
- `foldLeft [A] (z : B) (f : (B, A) => B) : B`
- `filter [A] (p : (A) => Boolean) : List[A]`

1. Write a non-recursive 1-2 line function `sevens (l : List[Int]) : List[Int]` that returns a list containing all of the integers in `l` with the value 7.

2. Write a non-recursive 1-2 line function `squares (l : List[Int]) : List[Int]` that squares each of the integers in `l`.

3. Write a non-recursive 1-2 line function `odd (l : List[Int]) : List[Int]` that returns a list containing the odd integers in `l`.

4. Write a non-recursive 1-2 line function `sum (l : List[Int]) : Int` that evaluates to the sum of the integers in `l`.

5. Write a non-recursive 1-2 line function `hasThree (l : List[Int]) : Boolean` that evaluates to true if and only if `l` contains the integer 3 at least once.

6. Write a non-recursive 1-2 line function `isBig (l : List[Int]) : List[Boolean]` that returns a list replacing each integer in `l` with true if the integer is larger than 10 and false otherwise.

Structural Induction

Consider the following function:

```scala
def append[T](xs: List[T], ys: List[T]): List[T] = xs match {
  case Nil => ys
  case x::xs => x::append(xs, ys)
}
```

We claim the `append` is associative. State this claim formally and prove it using structural induction.