Recitation Session, October 25 2017

Please do not write on this sheet of paper

And do not use laptops during the session

Structural Induction

Ex1. Prove that the following equivalence holds by using inductive reasoning:

```
alist map f map g === alist map (f andThen g)
```

Axioms:

```
1) Nil map f === Nil
2) (x :: xs) map f === f(x) :: (xs map f)
3) (f andThen g)(x) === g(f(x))
```

Note: Be *very* precise in your proof:

- Clearly state which axiom you use *at each step*, and when/if you use the induction hypothesis.
- Use only 1 axiom/hypothesis at each step, and only once. Applying 2 axioms requires 2 steps.
- Underline the part of an equation on which you apply your axiom.
- Make sure to state what you want to prove, and what your induction hypothesis is, if any.

Ex2. A more complicated proof (midterm 2016)

We want to implement a function `sum(list: List[Int]): Int`, which returns the sum of the elements of a list of `Int`s. We can easily *specify* that function as follows:

```
(1)  sum(Nil) === 0
(2)  sum(x :: xs) === x + sum(xs)
```

If we naively translate this specification into a Scala implementation, we end up with a uselessly non-tail recursive function. Besides, doing the recursion ourselves is wasteful. Instead, we implement it using `foldLeft`:

```
def betterSum(list: List[Int]): Int =
list.foldLeft(0)(add)

def add(a: Int, b: Int): Int = a + b
```

However, that implementation is not trivially correct anymore. We would like to *prove* that it is correct for all lists of integers. In other words, we want to prove that

```
list.foldLeft(0)(add) === sum(list)
```

for all lists of integers.

In addition to the specification of `sum` (1-2), you may use the following axioms:

```
(3)  Nil.foldLeft(z)(f) === z
(4)  (x :: xs).foldLeft(z)(f) === xs.foldLeft(f(z, x))(f)
(5)  add(a, b) === a + b
(6)  a + b === b + a
(7)  (a + b) + c === a + (b + c)
(8)  a + 0 === a
```

Axioms 3-5 follow from the implementations of `foldLeft` and `add`. Axioms 6-8 encode well-known properties of `Int.+`: commutativity, associativity, and neutral element.

Your task: prove the following lemma by structural induction:

```
list.foldLeft(z)(add) === z + sum(list)
```

From that lemma, we can "trivially" (with the help of axioms 6 and 8) derive that `betterSum`'s implementation is correct by substituting `0` for `z` in the lemma. You are not asked to do that last bit.