CS119 – Lab 2B
Due Date: February 20

Purpose: Induction and recursion are two sides of the same coin. Both need a base case and use a smaller solution to prove (or compute) a larger solution. Put all together you get a proof (or a computation) which will work for any size. Because of their interrelatedness, induction may be used to prove the correctness of recursive functions.

Knowledge: This lab will help you become familiar with the following content knowledge:

- How to write induction proofs which prove that our recursive functions work correctly for all inputs.

Task: Follow the steps in this lab carefully to complete the assignments.

As you go through the assignment, the base case for the induction is usually easy to check but don't forget this important step. Most of your work will be in proving an *if*. In other words, you will be assuming the existence of a smaller solution and using that to show the larger result. It may feel weird or that you are using circular reasoning by making the assumption of the smaller solution but you are not. You are simply proving an *if* statement is valid. You are arguing that *if* one thing is true then another thing must be true as well.

Consider the following example. Suppose I define a function: `g n = (f n) + 1`. Without even knowing what the function `f` is you should be able to show that this statement is valid: "if `(f n)` returns a value greater than 2 then `(g n)` returns a value greater than 3". You would simply assume that `(f n)` returns a value greater than 2 and then argue that it must follow that `(g n)` will return a value greater than 3. This same type of argument will be used for your induction steps.

Assignment 1:
Consider the function

```
f :: Int -> Int
f n = if n==0
        then 0
        else 1 + (f(n-1))
```

Use induction to show that the function `f` returns the value of n for all possible inputs $n \geq 0$.
Here are the steps:

1. Verify that `f 0` returns 0 to show the base case.
2. Show that *if* `f(n-1)` returns $n-1$ then `f n` returns n.
3. Since you have shown the base case and the induction step, you can confidently state that the function works for all possible nonnegative input values.

Hint: To show that "if `f(n-1)` returns $n-1$ then `f n` returns n" is valid you need to assume that `f(n-1)` returns $n-1$ and then argue that it must follow that `f n` returns n. Use the definition of the function and just a little bit of algebra.

Criteria for Success: You have clearly written down all three steps of the inductive proof. Your proof contains **complete sentences** which explain all the steps and the algebra. I don't want to see just a bunch of symbols on a page!

Assignment 2:
Consider the function

```
g :: Int -> Int
g n = if n==0
        then 0
        else 2 + (g(n-1))
```

Use induction to show that the function `g` returns the value of $2n$ for all possible inputs $n \geq 0$.

Criteria for Success: You have clearly written down all three steps of the inductive proof.

Assignment 3:
Consider the function

```
h :: Int -> Int
h n = if n==0
        then 5
        else 10 + (h(n-1))
```

Use induction to show that the function `h` returns the value of $5 + 10n$ for all possible inputs $n \geq 0$.

Criteria for Success: You have clearly written down all three steps of the inductive proof. Your proof contains **complete sentences** which explain all the steps and the algebra. I don't want to see just a bunch of symbols on a page!

Assignment 4:
Consider the function

```
revWord :: Language -> Language
revWord w = if (empty w)
        then w
        else (lastItem w) +++ revWord (butLast w)
```

Prove by induction that for a word of length ≥ 0, the function `revWord` returns the reverse of the word.

Hint: We are using induction on the length of the words. So you show the function will work correctly for words of length 0. Then show "If the function works correctly in reversing words of length $n-1$ then it will also work correctly in reversing words of length n".

Here are the steps:

1. Verify that `revWord` returns the correct result for the empty word, which is the base case.
2. Suppose we have a word of length n, which we will denote as $x_1x_2...x_{n-1}x_n$ for arbitrary characters x_i. Show that *if* `revWord` works for all words of length $n-1$ then `revWord` $x_1x_2...x_{n-1}x_n$ also works correctly.
3. Since you have shown the base case and the induction step, you can confidently state that the function works for all possible word input values.

Criteria for Success: You have clearly written down all three steps of the inductive proof. Your proof contains **complete sentences** which explain all the steps.

Submit your proof writeups in Canvas for grading.