CS119 – Module 3: Induction

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.

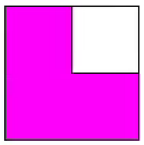
Knowledge: This module will help you become familiar with the following content knowledge:

- Induction to prove our recursive functions work correctly for all inputs

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 20 minutes.

1. A triomino is an L shaped tile. We want to know whether we can lay the triominos on a square board such that the triominos don't overlap and the entire board is covered except for one corner.

   Clearly we can do so for a 2 by 2 board with just one triomino.

   Try to tile a 4 by 4 board with triominoes. It will take 5 triominoes to do it.

2. Try to tile an 8 by 8 board.
   Hint: Take 4 of your 4 by 4 solutions and one extra triomino and arrange them for an 8 by 8 board. Consider where you want the blank corner to be on each of the 4 by 4 solutions. One will clearly be for the blank corner but the other three need to form a nice triomino shaped blank.

3. Now we want to prove that we can tile any board of size 2^n by 2^n by using induction. You have already proved the base case when n is 1 since we can tile 2^1 by 2^1 board. If you can tile a 2^{n-1} by 2^{n-1} board, explain how you can also tile a 2^n by 2^n board. The key to induction proofs is showing how having a smaller solution can give you a bigger solution. (If you think this sounds a lot like recursion, you are right!)

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

1. Consider the function `f`. We want to show that for **ALL** nonnegative inputs that `f n` returns `n+1`. We can't possibly try all the inputs so we need to use mathematical induction.

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

   (a) We start by verifying that the function works correctly for the base case. Does `f` return the correct value for the base case? Why is it correct?

   (b) The rest of the proof involves proving an *if*. In other words we need to show "If `f` works correctly for values less than `n` then it works correctly for `n` as well." Why would proving this *if* statement true give us the result that `f` is correct for all nonnegative inputs?

   (c) To prove this *if* we suppose that `f k` works correctly for $0 \leq k < n$. Then given that supposition we will show that `f n` works correctly. Why would that "prove the if?"

   (d) Using the definition of the function `f` we know that `f n = 1 + f(n-1)`. Since `n-1` is greater than equal to 0 and less than n, what do we know the correct value for `f (n-1)` must be?

   (e) Use the value of `f(n-1)` to get the value of `f n` and you have done it!

2. Your evil professor has written the following function and claims that it correctly squares any nonnegative input. Verify she is correct by using induction.

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

   (a) Is the base case correct?

   (b) If `square (n-1)` is correct (gives us $(n-1)^2$) then is `square n` correct (gives us n^2)? Why?

Complete the following assignments to be submitted for grading. Each should be done individually but you can consult with a classmate to discuss your strategies.

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. 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 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 written answers for grading in Canvas.