CISC106 Summer 2011 Lab05

- This lab an all subsequent labs will be due Sunday at 11:55 PM EDT on Sakai.
- a lab05_tests.py skeleton is provided for you on the course website.

Preparation (do not submit for grading)

1. Implement a version of `factorial` that works recursively rather than using a for loop. Which version of the function do you like better: this new recursive one or the original *iterative* (*i.e* with the loop) one?

Programs (to be graded)

1. Download lab05.py and Frames.py from the course website. Run lab05.py and you should get a window with Link from *The Legend of Zelda* somewhere near the middle. Try moving him around on the screen:

   - `w` will cause him to move *up*
   - `a` will cause him to move *to the left*
   - `s` will cause him to move *down*
   - `d` will cause him to move *to the right*

   If he reaches the edge of the window, he'll wrap around to the other side.

   By this point you should be noticing that there's a problem. The controls are all wrong! Since they're hard coded in the program, it's your job to figure out how to change the code to make them right.

2. Ackermann's function (named after mathematician Wilhelm Ackermann) is a recursively defined function with some properties of interest to theoretical computer science. The function is defined as such:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{otherwise} \end{cases} \quad (1)$$

   Write a function called `ackermann` inside lab05.py which implements Ackermann's function.

3. Now you should write a unit test for your implementation of Ackermann's function. You should include at least three assertEquals (see `http://en.wikipedia.org/wiki/Ackermann_function#Table_of_values` for a table of return values. You'll probably want to stick to $m \leq 3$ and $n \leq 6$ as input for the assertEquals.) and two *assertRaises*. A valid call to assertRaises may look like: [1]

```
self.assertRaises(TypeError, ackermann, 3, 4)
```

[1] N.B. This example assert will fail

What the above line does is test to see whether or not calling `ackermann`$(3, 4)$ raises a `TypeError`. Before you write your assertRaises, you should try calling ackermann at the python prompt (remember to `import lab05` first) with some large numbers to see what exception you get.

You should submit lab05.py and lab05_tests.py along with any other docs required by your TA on Sakai.