HW 6 **Prof. Caldwell**
Due: 9 September 2011 **COSC 3015**

Problem 0.1. If you haven't done so - read Chapter 1 (Starting Out) from LYAHFGG and read chapter 2 (Believe the Type).

Note that the Haskell function *error* takes a string and halts the computation using the string as the error message.

Prelude> error "Oops!"
*** Exception: Oops!*
Prelude> error "Splat!"
*** Exception: Splat!*

Problem 0.2. Write functions having the following types: You will need to use recursion - and don't just use the built-in version if there is one.

$$\begin{array}{lll} last & :: & [a] \to a \\ select & :: & [a] \to Int \to a \\ middle & :: & [a] \to a \\ split & :: & [a] \to Int \to ([a],[a]) \\ repeat & :: & (a \to a) \to Int \to a \to a \end{array}$$

The *last* function takes a list and returns the last element of the list or calls *error* if the list is empty. The function *select* takes a list (say xs) and an integer (say k) and returns the k^{th} element of the list xs (using zero based indexing). If $k < 0$ or $k \geq$ *length* xs then call *error*. The *middle* function takes a list and returns the middle element – if the list is of even length, you can implement your function to have a leftist or rightist bias – your choice. emsplit take s a list and a position k in the list and returns a pair of lists. The first element of the pair contains the first k elements of the input list and the second element of the pair contains the the $k + 1^{st}$ through the last element of the list. It should be that if you append the two output lists - you get back a list equal to the input list. A call *repeat* f k x applies the function f k times to the input x. Thus

$$repeat\ f\ k\ x = \overbrace{f(f \cdots (f}^{k\ times}\ x))$$

If $k = 0$ then the result is the identity function *id*.

You should implement some tests to convince the grader your code works.