CS 5035 (Fall 2016)

### Project 7a. Making our own types and typeclasses.

Based on chapter [8 of LYH](http://learnyouahaskell.com/making-our-own-types-and-typeclasses). [Videos](https://sites.google.com/a/lclark.edu/drake/courses/pls/lesson-7-haskell-making-our-own-types-and-typeclasses).

Here is a program that plays [Tic-Tac-Toe](https://drive.google.com/file/d/0B-I58s-_d3o5bTRRbzlQX1p5NEU/view?usp=sharing). It is based on a program by Peter Drake. Download it, load it into GHCi, play a few games, and follow along in the code.

Play a game by using the following commands.

Start the game (‘X’ always goes first) with either start or startAt r c.

> start -- Start a game by placing an X at row 1, col 1.

> startAt r c -- Start a game by placing an X at row r, col c.

Make a move after the first move with either next it or play r c it.

> next it -- Place an X or an O (depending on whose turn it is)

-- optimally.

> play r c it -- Place an X or an O (depending on whose turn it is)

-- at row r, col c.

The game takes advantage of the fact that the Haskell REPL uses it to refer to the most recent value. So the it in the commands above refer to the result of the previous play. A command that uses it must follow immediately after a previous move in the game.

Your responsibility is to read the code and explain it to me. For this assignment you should understand and be able to explain:

* the type and constant declarations: lines 8 – 33;
* in particular, how the Game data type is created (line 22) and used throughout; (Game is very much like a **class** in Java.)
* the type declarations for the other functions, including the **where** sub-functions;
* the intent of the functions, i.e., what they claim to compute. For now, you need not understand the details of how those computations are performed. A good way to explain the intent of a function is to generate some example inputs and outputs and to discuss the relationship between a function’s input and the corresponding output.
* how the Maybe data type is used throughout.

***Hint****: the code is written to drag out a game as long as possible without giving up an advantage. So an “optimal” move by* next *may not be what you expect.*