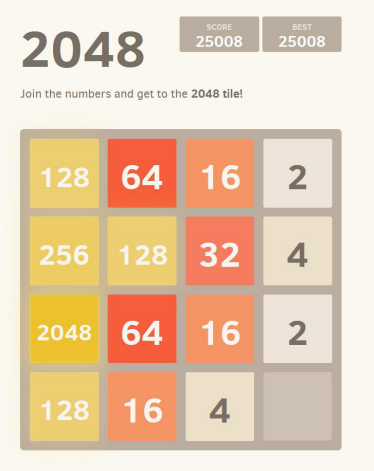
COMP 524 – Spring 2014
Extra Credit Assignment
Haskell 2048 Game

Due: 4/25/14 by 11:59pm via email to bcw@cs.unc.edu

In this extra credit assignment, you will have the chance to apply functional programming techniques in Haskell to develop a complete program for the popular game of 2048. We will break the game up into several parts and you may complete as many or as few as you would like. Extra credit points will be awarded accordingly.

2048 Game Description

The game of 2048 is a one-person game played over a 4x4 grid, or board, of integer tiles. The game works by sliding all of the tiles on the board left, right, up or down. When two tiles of the same value collide, they will merge into a single tile with a value equal to the sum of the two colliding tiles. In each turn, or slide, all the tiles on the board slide as far as possible in the indicated direction. After sliding, a new tile with the value of two or four will appear in an unoccupied random tile on the board. The objective of the game is to merge enough tiles together until at least one tile has a value of 2048 (as seen at the right). However, you lose if the entire board is full and you cannot merge any tiles by sliding in any direction. To get a better understanding of the game, you can try it online at `http://gabrielecirulli.github.io/2048/`.



In this assignment, we will break up the functionality of the game into multiple parts, each graded separately for extra credit points. If you build all of the parts, you will have a working Haskell 2048 game. Note that in implementing each part, you may find it useful to develop helper functions to make your code more modular and readable (and it will be easier to develop too!).

1 Slide Left (15 points)

Develop a function that takes as input a game state (the type of this data structure is left to your discretion), and returns a game state after a left slide. If a left slide is not possible (all of the tiles are already as far left as possible, and no merges are possible), you may return the same board.

2 Slide (10 points)

Once you have developed a left slide function, develop a more general slide function that takes as arguments a board and a slide direction, and then applies the appropriate slide. (Hint: you may want to call your left slide function in the implementation of the other sliding directions.)

3 Game Over Check (10 points)

The game of 2048 is over when the entire board is full, and a slide in any direction will not cause any merges to free up any new spaces. Implement a function to check whether a given game state is over or not. We do not assume that the game is over if a 2048 tile is constructed, we will let the user play for 4096 or higher.

4 Adding a new tile (30 points)

After each slide or turn in 2048, a new tile is placed in a random location on the board. 90% of new tiles are 2's while the other 10% are 4's. Develop a function that implements this functionality.

In this function, you will need to make use of random numbers. This functionality can be implemented using techniques described in Chapter 9 of our Haskell book (though other approaches exist and are permissible). Additionally, I strongly encourage you to decompose this function and develop several helper functions. For example, you may find a function useful to find all of the board locations that are empty, choose a random tile value, or update an entry on the board, among other things.

5 Main function (35 points)

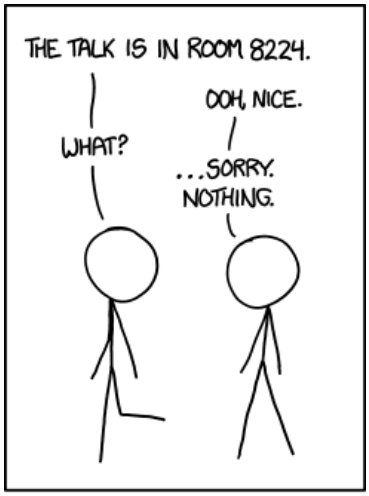
In this last part, you will implement a main function, and perhaps helper functions, that integrate all of the previous parts into a playable command-line-based 2048 game. For each turn, print the board and read from standard input a slide direction (*e.g.*, "left", "right", "up", "down", or your favorite hotkeys). If the slide direction is valid, slide the board and add a new tile to the board. If the game is over, terminate, otherwise, play the next turn.

Deliverables

Along with your Haskell source file(s), you must submit a text file that describes which functions you implemented for each part. Give an example invocation of each function (excluding helper functions), or, in the case of the last part, describe how to compile and/or run your program.

Honor Code

You may work in pairs on this extra credit assignment. You may discuss the assignment with other pairs, but you **cannot** share source code. Additionally, you may make use of any standard Haskell libraries.



http://xkcd.com/1344/