CSC 435 Spring 2014

Assignment 4 – Explore Haskell

Due: March 21, 2014 by 11:59 p.m.
Grade: 60 points

Objective:

The objective of this assignment is for students to get a deeper understanding of the functional programming paradigm by implementing a problem solution using Haskell.

Requirements:

You will complete this assignment individually.

Review the two projects described below and select either **one** to implement.

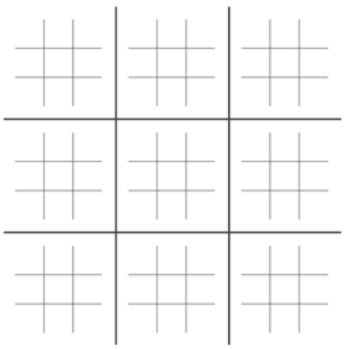
Your program must include the following.

- The players should enter their desired moves via the command line.
- Error detection should be employed to only allow valid moves to be made.
- Terminal states should be detected to determine when the game is over.
- The board state should be displayed and formatted nicely in the console after each move.
- The specific strengths of Haskell should be utilized.

Project 1: Ultimate Tic-Tac-Toe (designed by Haskell Group A).

Objective:
The objective of this project is to implement a playable game of Ultimate Tic-Tac-Toe in Haskell.

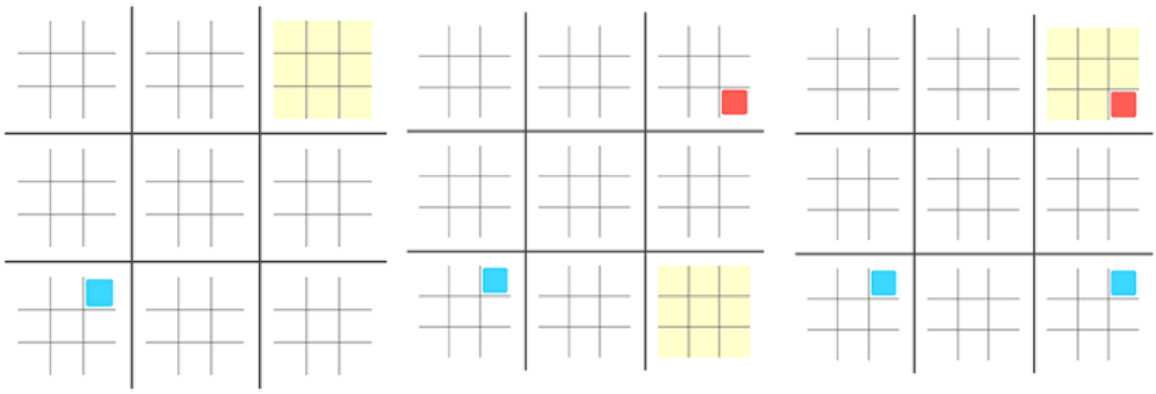
Details:
Ultimate Tic-Tac-Toe is a twist on the classic game of tic-tac-toe. The playing board starts out as a three by three grid of empty tic-tac-toe boards as illustrated in the following image:



The game of ultimate tic-tac-toe adheres to the following rules:

- In each turn, the player marks one of the small squares.
- When a player gets three in a row on a small board, the player wins that board.
- To win the game, a player needs to win three small boards in a row.
- A player may only play on the board in the big square that corresponds to the last small square the opponent played on.
- If the board the player must play on is already won, the player may choose to play on any incomplete board.

For example, the following image shows a valid sequence of moves:



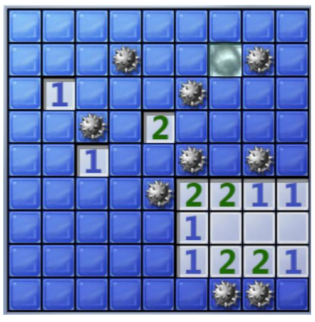
Project 2: Dustsweeper (designed by Haskell Group B).

Objective:
The objective of this project is to implement a program in Haskell that plays a game of "Dustsweeper".

Details:
Dustsweeper is a variation on the game Minesweeper. In this game, the player must sweep away all the dust that is under each rug. However if a player steps on a rug with dust under it the player slips and falls, and thus loses the game.

A board is created with each cell representing a "rug".

- The initial board has a certain number of dust balls, which are hidden under rugs and so not seen by the user.
- When a player selects a cell, the dust ball is exposed, if present.
- If the player selects a rug that does not contain a dust ball, a number will be displayed that indicates the number of dust balls adjacent to the current square. In the example below, the "1" squares each have only one dust ball next to them, while the two "2", squares each have two dust balls adjacent to them.



Implement a program that allows two players to play a Dustsweeper game given a board. Dustsweeper boards can be of any size as long as both the height and width are equal.

Rubric:

Implements a fully working solution to the problem. This means that the code compiles and the game is played as expected according to the rules.	20 points
Uses functional programming concepts, and leverages the strengths of Haskell, such as type classes, monads, pure functions, and lazy evaluation.	20 points
Handles console input / output aesthetically and performs appropriate error checking.	10 points
Clean and clear, well-documented, logical code	10 points

Bonus 10 points: Add some AI to your program so that the second player is the computer instead of a human.

Deliverables:
Submit on Canvas in the "Assignment 4" category a zip file containing:

- Source code file(s).
- Script file showing successful execution.
- All input files used and output files generated.
- A readme.txt (text) file specifying which project has been implemented, and instructions for running the program successfully. Also clearly indicate whether the bonus part has been attempted.

Grade:

As per the rubric.