Assignment 03 (Due: Monday, November 7, 2016, 11 : 59 : 00PM (Central Time))

CSCE 322

Contents

List of Figures

1 Instructions

In this assignment, you will be required to write Haskell functions that simplify playing of the variation of Greater Than Sudoku.

1.1 Data File Specification

An example of properly formatted file is shown in Figure 1. The file encodes a tuple containing list of spaces to place values into, the values to be placed in the corresponding spaces, the visible grid, the inequalities between the rows (-1 means decreasing from top to bottom, 1 means increasing from top to bottom), and the inequalities between the columns (-1 means decreasing from left to right, 1 means increasing from left to right).

part01test01.gts

```
(
[16,8,6,9,14,13,3,10,14,7,6,5],
[2,4,1,1,3,3,3,1,4,3,3,1],
[
"4---",
"3---",
"--24",
"2-3-"
],
[
[-1,-1,1,-1],
[-1,1,-1,1],
[1,1,1,-1]
],
[
[-1,-1,1],
[-1,1,-1],
[1,-1,1],
[1,-1,-1]
]
)
```

Figure 1: A properly formatted game encoding

Spaces are defined from the upper-left corner of the game (Position 1), down the first column, down the second column... to the bottom of the last column.

1.2 `csce322hw03pt(num).hs`

This assignment requires the implementation of two (2) methods: `onePlayerOneMove` and `onePlayerManyMoves`. Each method will be implemented in its own file (named `csce322hw03pt(num).hs`, where `(num)` is 01 or 02). The behavior of each function is described below.

1.2.1 `onePlayerOneMove (csce322hw03pt01.hs)`

`onePlayerOneMove :: [[Char]] -> Int -> Int -> [[Char]]` This method will take a move to make and a game, and return a game after the move according to the following rules:

1. If the space is already full, no change is made to the game
2. If the move would violate a rule immediately (including a 1 to the left of a $>$ sign or a *numRows* to the left of a $<$ sign), no change is made to the game
3. The value is placed in the correspond space, otherwise.

```
(
[16,8,6,9,14,13,3,10,14,7,6,5],
[2,4,1,1,3,3,3,1,4,3,3,1],
[
"4---",
"3---",
"--24",
"2-3-"
],
[
[-1,-1,1,-1],
[-1,1,-1,1],
[1,1,1,-1]
],
[
[-1,-1,1],
[-1,1,-1],
[1,-1,1],
[1,-1,-1]
]
)
```

Figure 2: Game State Before `onePlayerOneMove`

```
"Result"
"4---"
"3---"
"--24"
"2-3-"
""
```

Figure 3: Game State After `onePlayerOneMove`

Sample Sequence

1.2.2 `onePlayerManyMoves (csce322hw03pt02.hs)`

```
onePlayerManyMoves :: [[Char]] -> [Int] -> [Int] -> [[Char]]
```

This method will take moves to make and a game, and return a game. If a move is invalid, it is skipped and the next move is attempted. The same rules as `onePlayerOneMove` apply.

```
(
[16,8],
[2,4],
[
"4---",
"3---",
"--24",
"2-3-"
],
[
[-1,-1,1,-1],
[-1,1,-1,1],
[1,1,1,-1]
],
[
[-1,-1,1],
[-1,1,-1],
[1,-1,1],
[1,-1,-1]
]
)
```

Figure 4: Game State Before `onePlayerManyMoves`

```
"Result"
"4---"
"3---"
"--24"
"243-"
""
```

Figure 5: Game State After `onePlayerManyMoves`

Sample Sequence

2 Naming Conventions

You will be submitting at least 2 `.hs` files (`csce322hw03pt01.hs` and `csce322hw03pt02.hs`). If you do not submit a modified `Helpers.hs` file, the default one will be provided.

3 webgrader Note

Submissions will be tested with `ghc`. `cse.unl.edu` is currently running version 7.6.1 of `ghc`. If you would like to test things offline, you can load a file in `ghci` with the command `:l filename.hs` and run the main method on a given file with the command `:main "/path/to/inputfile.cnf"`

4 Point Allocation

Component	Points
`csce322hw03pt01.hs`	50%
`csce322hw03pt02.hs`	50%
Total	100

5 External Resources

Learn Haskell Fast and Hard
Learn You a Haskell for Great Good!
Red Bean Software
Functional Programming Fundamentals The Haskell Cheatsheet