Assignment 03 (Due: Friday, November 14, 2014)

CSCE 322

Contents

List of Figures

1 Instructions

In this assignment, you will be required to write Haskell functions that facilitate the playing of Chutes & Ladders. You will be provided with skeleton code that includes functionality for reading from a data file and generating outputs. The code will also include function declarations that you must define.

1.1 Data File Specification

An example of a properly formatted file is shown in Figure 1. The tuple represents the positions of players, locations of chutes, locations of ladders, the board and die rolls to be processed.

```
(
[13,30,22,33],
[31,46,44],
[3],
[
[44,22,46,36,48,49,0],
[43,42,41,40,39,38,37],
[30,31,2,33,34,35,36],
[29,28,27,26,25,24,23],
[16,17,18,19,20,21,22],
[15,14,13,12,11,10,9],
[2,3,42,5,6,7,8]
],
[6,6,5,1,6,4,1,2,5,6,6,1,3,4,3,4,1,4,3,3,3,4,6,3,5,6,1,1,2,5]
)
```

Figure 1: A properly formatted Chutes & Ladders encoding

1.2 `csce322as03pt(num).hs`

This assignment requires the implementation of four (4) methods: `onePlayerOneMove` , `onePlayerManyMoves` , `manyPlayersManyMoves` , and `makeBoard` . Each method will be implemented in its own file (named `csce322as03pt0(num).hs`, where `(num)` is 1, 2, 3, or 4). The behavior of each function is described below.

1.2.1 `onePlayerOneMove (csce322as03pt01.hs)`

`onePlayerOneMove :: (Int,[Int],[Int],[[Int]],Int) -> Int` This method will take a player position, location of chutes, location of ladders, board, and die roll and change the player position based on these rules:

1. If the player will move past the last position on the board, the move is not made and the player's current position is returned
2. If the player will land on a chute or ladder, the player will be moved to the position where the chute/ladder ends
3. Otherwise, the player will move forward by the amount determined by the roll

```
(
[13,30,22,33],
[31,46,44],
[3],
[
[44,22,46,36,48,49,0],
[43,42,41,40,39,38,37],
[30,31,2,33,34,35,36],
[29,28,27,26,25,24,23],
[16,17,18,19,20,21,22],
[15,14,13,12,11,10,9],
[2,3,42,5,6,7,8]
],
[6,6,5,1,6,4,1,2,5,6,6,1,3,4,3,4,1,4,3,3,3,4,6,3,5,6,1,1,2,5]
)
```

Figure 2: Game State Before `onePlayerOneMove`

```
"Positions"
[19]
```

Figure 3: Result After `onePlayerOneMove`

1.2.2 `onePlayerManyMoves (csce322as03pt02.hs)`

`onePlayerManyMoves :: (Int,[Int],[Int],[[Int]],[Int]) -> Int` This method will return the position that a single player will end on after a series of die rolls (following the same rules as `onePlayerOneMove` .

```
(
[208,265],
[127,211,247],
[82],
[
[0,288,287,286,285,284,283,282,281,280,279,278,277,276,275,274,273,272],
[254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271],
[253,252,251,250,249,154,247,246,245,244,243,242,241,240,239,238,237,236],
[218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235],
[217,216,215,214,213,186,211,210,209,208,207,206,205,204,203,202,201,200],
[182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199],
[181,180,179,178,177,176,175,174,173,172,171,170,169,168,167,166,165,164],
[146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163],
[145,144,143,142,141,140,139,138,137,136,135,134,133,132,131,130,129,40],
[110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127],
[109,108,107,106,105,104,103,102,101,100,99,98,97,96,95,94,93,92],
[74,75,76,77,78,79,80,81,82,171,84,85,86,87,88,89,90,91],
[73,72,71,70,69,68,67,66,65,64,63,62,61,60,59,58,57,56],
[38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55],
[37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,22,21,20],
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
],
[6,5,4,3,4,1,5,6,3,2,4,1,6,6,3,3,5,1,2,2,5,6,2,3,4,2,1,4,5,3]
)
```

Figure 4: Game State Before `onePlayerManyMoves`

```
"Positions"
[222]
```

Figure 5: Result After `onePlayerManyMoves`

1.2.3 `manyPlayersManyMoves (csce322a03p03.hs)`

`manyPlayersManyMoves :: ([Int],[Int],[Int],[[Int]],[Int]) -> [Int]` This method will take a list of positions (as opposed to just one) and move the players (starting with the player in the first position) according to the die rolls (one roll per player, cycling around to the first player until the rolls have been exhausted or one player has reached the final position on the board). The rules of `onePlayerOneMove` will be followed.

```
(
[50,34,43,2],
[58,42,44,52],
[25],
[
[0,70,69,68,67],
[62,63,64,65,66],
[61,60,4,58,57],
[52,16,54,55,56],
[51,50,49,48,47],
[42,14,44,12,46],
[41,40,39,38,37],
[32,33,34,35,36],
[31,30,29,28,27],
[22,23,24,25,54],
[21,20,19,18,17],
[12,13,14,15,16],
[11,10,9,8,7],
[2,3,4,5,6]
],
[4,1,3,2,2,3,6,5,3,5,2,5,3,5,6,1,1,5,1,5,2,2,1,3,4,6,5,3,3,6]
)
```

Figure 6: Game State Before `manyPlayersManyMoves`

```
"Positions"
[69,67,60,26]
```

Figure 7: Result After `manyPlayersManyMoves`

1.2.4 `makeBoard (csce322a03p04.hs)`

`makeBoard :: (Int,Int,[Int],[Int]) -> [[Int]]` This method will take a number of rows and columns, along with locations of chutes and ladders, and return a board that satisfies those requirements.

```
(
[18,21],
[29,30],
[7],
[
[38,39,40,41,42,43,44,45,0],
[37,36,35,34,33,32,20,19,29],
[20,21,22,23,24,25,26,27,28],
[19,18,17,16,15,14,13,12,11],
[2,3,4,5,6,7,19,9,10]
],
[5,4,3,5,1,5,5,4,5,2,1,2,2,1,1,2,4,2,5,2,3,3,6,1,3,2,4,5,4,1]
)
```

Figure 8: Game State Before `makeBoard`

```
"Board"
[38,39,40,41,42,43,44,45,0]
[37,36,35,34,33,32,1,1,29]
[20,21,22,23,24,25,26,27,28]
[19,18,17,16,15,14,13,12,11]
[2,3,4,5,6,7,16,9,10]
""
```

Figure 9: Result After `makeBoard`

1.2.5 `manyPlayersManyMovesSixes (csce322a03p05.hs) (15 Points Extra Credit)`

`manyPlayersManyMovesSixes :: ([Int],[Int],[Int],[[Int]],[Int]) -> [Int]` In some variations of Chutes & Ladders, rolling three (3) sixes in a row results in the offending player being sent back to the first position. This method will enforce those rules. Otherwise, the game will proceed as `manyPlayersManyMovesSixes` does.

1.3 README.txt

This file should contain any assumptions that you made and sources that you used during the completion of this assignment.

2 Naming Conventions

You will be submitting 4 or 5 `.hs` files and 1 `README.txt` file. Because Web Handin will not allow more than 5 files to be submitted at once, you must submit extra files in an additional upload. The files will still show up under the same submission.

Component	Points
`csce322as03pt01.hs`	25
`csce322as03pt02.hs`	10
`csce322as03pt03.hs`	30
`csce322as03pt04.hs`	25
`README.txt`	10
Total	100

3 Point Allocation

4 External Resources

Learn Haskell Fast and Hard
Learn You a Haskell for Great Good!
Red Bean Software
Functional Programming Fundamentals
The Haskell Cheatsheet
Category:Haskell - Rosetta Code
Haskell - Wikibooks