Assignment 03 (Due: Monday, November 6, 2017, 11 : 59 : 00PM Central Time)

CSCE 322

1 Instructions

In this assignment, you will be required to write Haskell functions that simplify navigating an elaborate maze.

1.1 Data File Specification

An example of properly formatted file is shown in Figure 1.

part01test01.emf

```
(
"llulrruul",
[
"xxxxxxxxxx",
"xxx-----1x",
"xx--x----x",
"x---x--xxx",
"x-x-x-xx-x",
"x-x------x",
"x-----xx-x",
"x--xg--x-x",
"xx-x-----x",
"xxxxxxxxxx"
]
)
```

Figure 1: A properly formatted maze encoding

2 One Player, One Move

The first part (`onePlayerOneMove` in the file `csce322homeWork03part01.hs`) will take in two (2) arguments (the maze and a move) and returns one (1) value (the maze that is the result of Player 1 making the given move) . If the maze has already been solved, or the direction of the move is immediately blocked, the maze is unchanged. Otherwise, like the JavaScript assignment, the player keeps moving until the goal is reached, they reach a dead end (where the only open

move is “backwards”), or they occupy a spot with more options than just moving “forward” and “backward”.

```
(
"llulrruul",
[
"xxxxxxxxxx",
"xxx-----1x",
"xx--x----x",
"x---x--xxx",
"x-x-x-xx-x",
"x-x------x",
"x-----xx-x",
"x--xg--x-x",
"xx-x-----x",
"xxxxxxxxxx"
]
)
```

Figure 2: Before `onePlayerOneMove`

```
"Result"
"xxxxxxxxxx"
"xxx----1-x"
"xx--x----x"
"x---x--xxx"
"x-x-x-xx-x"
"x-x------x"
"x-----xx-x"
"x--xg--x-x"
"xx-x-----x"
"xxxxxxxxxx"
""
```

Figure 3: After `onePlayerOneMove`

3 One Player, Many Moves

The second part (`onePlayerManyMoves` in the file `csce322homeWork03part02.hs`) will take in two (2) arguments (the maze and a list of moves) and returns one (1) value (the maze that is the result of Player 1 making all of the give moves). If the maze has already been solved the maze is unchanged. If the direction of a move is immediately blocked, the maze is unchanged for that move.

```
(
"rllldlrdlrdul",
[
"xxxxxxxxxxxxxx",
"x-xxx---x---xxx",
"x--xx-x-x-x-xxx",
"xx--x-------x-x",
"xxxgxxxx-x-x--x",
"x-x-x--x--xx-xx",
"x-x----------xx",
"x--x1xx-x--x--x",
"x-------xx-x-xx",
"xxxxxxxxxxxxxx"
]
)
```

Figure 4: Before `onePlayerManyMoves`

```
"Result"
"xxxxxxxxxxxxxx"
"x-xxx---x---xxx"
"x--xx-x-x-x-xxx"
"xx--x-------x-x"
"xxx1xxxx-x-x--x"
"x-x-x--x--xx-xx"
"x-x----------xx"
"x--x-xx-x--x--x"
"x-------xx-x-xx"
"xxxxxxxxxxxxxx"
""
```

Figure 5: After `onePlayerManyMoves`

4 Many Players , One Move

The third part (`manyPlayersOneMove` in the file `csce322homeWork03part03.hs`) will take in two (2) arguments (the maze and a move) and returns one (1) value (the maze that is the result of Player 1 making the given move). If the maze has already been solved, or the direction of the move is immediately blocked, the maze is unchanged. This differs from the first part in that there may be more than 1 player in the maze.

```
(
"lr",
[
"xxxxxxxxxxx",
"x-------x--x",
"x-x-x-x----x",
"x---3--x--xx",
"xx----xxx--x",
"xx-xxx-2-x-x",
"xx-----x---x",
"xxx--xxxxx-x",
"x-x---1-xx-x",
"x-xx-xxxxx-x",
"x-g---x----x",
"xxxxxxxxxxx"
]
)
```

Figure 6: Before `manyPlayersOneMove`

```
"Result"
"xxxxxxxxxxx"
"x-------x--x"
"x-x-x-x----x"
"x---3--x--xx"
"xx----xxx--x"
"xx-xxx-2-x-x"
"xx-----x---x"
"xxx--xxxxx-x"
"x-x-1---xx-x"
"x-xx-xxxxx-x"
"x-g---x----x"
"xxxxxxxxxxx"
""
```

Figure 7: After `manyPlayersOneMove`

5 Many Players , Many Moves

The fourth part (`manyPlayersManyMoves` in the file `csce322homeWork03part04.hs`) will take in two (2) arguments (the maze and a list of moves) and returns one (1) value (the maze that is the result of Player 1 making the first move, Player 2 making the second...). If the maze has already been solved, or the direction of the move is immediately blocked, the maze is unchanged and the next player makes the next move.

```
(
"d",
[
"xxxxxxxxxxxx",
"xx-x--x----1x",
"xx-------x--x",
"x-2-x--x-x-xx",
"x-x-xxx--x--x",
"x--4---x--x-x",
"xxx3-xx--x--x",
"x-------x---x",
"xg------x-xxx",
"xxxxxxxxxxxx"
]
)
```

Figure 8: Before `manyPlayersManyMoves`

```
"Result"
"xxxxxxxxxxxx"
"xx-x--x-----x"
"xx-------x1-x"
"x-2-x--x-x-xx"
"x-x-xxx--x--x"
"x--4---x--x-x"
"xxx3-xx--x--x"
"x-------x---x"
"xg------x-xxx"
"xxxxxxxxxxxx"
""
```

Figure 9: After `manyPlayersManyMoves`

6 Naming Conventions

Your files should follow the naming convention of `csce322homeWork03part01.hs`, `csce322homeWork03part02.hs`, `csce322homeWork03part03.hs`, and `csce322homeWork03part04.hs`.

6.1 `Helpers.hs`

A file named `Helpers.hs` has been provided with the functionality to read the `.emf` files into matrices. If a modified `Helpers.hs` file is not included with your submission, the default will be used in its place.

7 webgrader Note

Submissions will be tested with `ghc`. `cse.unl.edu` is currently running version 7.10.3 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.bff"`

8 Point Allocation

Component	Points
`csce322homeWork03part01.hs`	
Compilation	10
Test Cases	1×10
Total	20
`csce322homeWork03part02.hs`	
Compilation	10
Test Cases	1×10
Total	20
`csce322homeWork03part03.hs`	
Compilation	10
Test Cases	1×20
Total	30
`csce322homeWork03part04.hs`	
Compilation	10
Test Cases	1×20
Total	30
Total	100

9 External Resources

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