Assignment 03 (Due: Monday, March 30, 2020, 11 : 59 : 00PM Central Time)

CSCE 322

1 Instructions

In this assignment, you will be required to write Haskell functions that simplify playing of the variation of Peg Solitaire.

1.1 Data File Specification

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

test03.eps

```
(
[
"xxx3-x-xxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxx2-xxxxx",
"xxxxx-x-xx",
"xxxx-4-x1x",
"xxx-xx-x-x",
"xxxxxxx-xx"
]
,
"urludlrrurrdrudrrrlu"
)
```

Figure 1: A properly formatted encoding

2 One Player, One Move

The first part (`onePlayerOneMove` in the file `csce322assignment03part01.hs`) will take in two (2) arguments (a game and a move) and returns the game that is the result of moving an `x` from the given direction into the place occupied by Player 1. An example is provided below

test03.eps

```
(
```

```
[
"xxx3-x-xxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxx2-xxxxx",
"xxxxx-x-xx",
"xxxx-4-x1x",
"xxx-xx-x-x",
"xxxxxxx-xx"
]
,
"urludlrrurrdrudrrrlu"
)
```

test03.onePlayerOneMove.solution

```
"Result"
"xxx--x-xxx"
"xxxxxxx-xx"
"xxxxxxxxxx"
"xxxxxxx-xx"
"xxxxxxxxxx"
"xxx--xxx1x"
"xxxxx-x--x"
"xxxx---xxx"
"xxx-xx-x-x"
"xxxxxxx-xx"
""
```

3 One Player, Many Moves

The second part (`onePlayerManyMoves` in the file `csce322assignment03part02.hs`) will take in two (2) arguments (a game and a list of moves) and returns the game that is the result of Player 1 playing each move in succession (following the rules of onePlayerOneMove) until all of the moves in the array have been played, only one (1) `x` remains in the game, or Player 1 cannot make any more valid moves.

test03.eps

```
(
[
"xxx3-x-xxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxx2-xxxxx",
```

```
"xxxxx-x-xx",
"xxxx-4-x1x",
"xxx-xx-x-x",
"xxxxxxx-xx"
]
,
"urludlrrurrdrudrrrlu"
)
```

test03.onePlayerManyMoves.solution

```
"Result"
"xxx--x-xxx"
"xxxxx-x-xx"
"xxxx-x-xxx"
"xxxxx-1-xx"
"xxxxxx-xxx"
"xxx--xx-xx"
"xxxxx-x--x"
"xxxx---xxx"
"xxx-xx-x-x"
"xxxxxxx-xx"
""
```

4 Many Players , One Move

The third part (`manyPlayersOneMove` in the file `csce322assignment03part03.hs`) will take in two (2) arguments (a game and a list of moves) and returns the game that is the result of each player in the game attempting to make exactly one move until each player has attempted a move or only one (1) `x` remains in the game. Player 1 attempts the first move, Player 2 attempts the second, etc. (you may assume the highest number in the provided game is the number of players in the game). The moves are made in the order they appear in the moves array.

test03.eps

```
(
[
"xxx3-x-xxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxx2-xxxxx",
"xxxxx-x-xx",
"xxxx-4-x1x",
"xxx-xx-x-x",
"xxxxxxx-xx"
]
,
```

```
"urludlrrurrdrudrrrlu"
)
```

test03.manyPlayersOneMove.solution

```
"Result"
"x3-x-x-xxx"
"xxxxxxx-xx"
"xxxxxxxxxx"
"xxxxxxx-xx"
"xxxxxxxxxx"
"xxx2-xxx1x"
"xxxxx-x--x"
"xxxx-4-xxx"
"xxx-xx-x-x"
"xxxxxxx-xx"
""
```

5 Many Players , Many Moves

The fourth part (`manyPlayersManyMoves` in the file `csce322assignment03part04.hs`) will take in two (2) arguments (a game and a list of moves) and returns the game that is the result of each player in the game taking turns making a move until all of the moves in the array have been exhausted or a player has won the game (left one `x` in the game). Player 1 attempts the first move, Player 2 attempts the second move, etc. (you may assume the highest number in the provided game is the number of players in the game).

test03.eps

```
(
[
"xxx3-x-xxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxxxxxx-xx",
"xxxxxxxxxx",
"xxx2-xxxxx",
"xxxxx-x-xx",
"xxxx-4-x1x",
"xxx-xx-x-x",
"xxxxxxx-xx"
]
,
"urludlrrurrdrudrrrlu"
)
```

test03.manyPlayersManyMoves.solution

```
"Result"
"xx-x-x-xxx"
```

```
"x-xxxxx-xx"
"x3xxxxxxx"
"xx-2xxx-1x"
"x-xxxxx-x"
"xx-x-xxxxx"
"xxxxx-x--x"
"xxxx-x-xxx"
"xxx-x--x-x"
"xxxxx4x-xx"
""
```

6 Extra Credit (10%)

Games will contain an arbitrary number of players.

7 Naming Conventions

Your files should follow the naming convention of `csce322assignment03part01.hs`, `csce322assignment03part02.hs`, `csce322assignment03part03.hs`, and `csce322assignment03part04.hs`.

7.1 `Helpers.hs`

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

8 webgrader Note

Submissions will be tested with `ghc`. `cse.unl.edu` is currently running version 8.0.2 of `ghc`.

9 Local Testing

```
ghc -o test.out csce322a03part0#.hs; ./test.out part01test01.eps
   ghci :l csce322a03part0#.hs :main part01test01.eps
```

10 Point Allocation

11 External Resources

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

Component	Points
`csce322assignment03part01.hs`	
Compilation	10
Test Cases	1×10
Total	20
`csce322assignment03part02.hs`	
Compilation	10
Test Cases	1×10
Total	20
`csce322assignment03part03.hs`	
Compilation	10
Test Cases	1×20
Total	30
`csce322assignment03part04.hs`	
Compilation	10
Test Cases	1×20
Total	30
Total	100