Assignment 02 (Due: Friday, March 13, 2020, 11 : 59 : 00PM Central Time)

CSCE 322

THIS ASSIGNMENT IS ONLY WORTH 10% OF YOUR FINAL GRADE.

1 Instructions

In this assignment, you will be required to write JavaScript 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. The first file encodes a game and the second file encodes the moves to be made.

part01test01.game.eps

```
x,x,x,x,x,x
x,x,-,x,-,x
x,-,x,-,x,-
x,x,-,x,x,1
x,x,x,x,x,x
x,x,x,x,x,x
x,x,x,x,x,x
```

part01test01.moves.eps

```
d,d,r,r,d,d,r,d,u,r,u,u,r,l,u,l
```

Figure 1: A properly formatted encoding

2 One Player, One Move

The first part (`onePlayerOneMove` in the file `csce322a02part01.js`) will take in one (1) argument (a game) and return a function that takes in one (1) argument (a move), and returns the game that is the result of Player 1 moving an `x` into the player's space in a given direction. An example is provided below

part01test01.game.eps

```
x,x,x,x,x,x
x,x,-,x,-,x
x,-,x,-,x,-
x,x,-,x,x,1
x,x,x,x,x,x
x,x,x,x,x,x
x,x,x,x,x,x
```

part01test01.moves.eps

```
d,d,r,r,d,d,r,d,u,r,u,u,r,l,u,l
```

part01test01.solution

```
game
[ 'x', 'x', 'x', 'x', 'x', 'x' ]
[ 'x', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', '-', 'x', '-' ]
[ 'x', 'x', '-', 'x', 'x', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', '-' ]
[ 'x', 'x', 'x', 'x', 'x', '1' ]
[ 'x', 'x', 'x', 'x', 'x', 'x' ]
```

3 One Player, Many Moves

The second part (`onePlayerManyMoves` in the file `csce322a02part02.js`) will take in one (1) argument (a game) and return a function that takes in one (1) argument (an arry 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, or Player 1 has no valid moves remaining (no `x` can jump over another `x` to land in the space occupied by `1`).

part02test01.game.eps

```
x,x,x,x,x,x
x,x,x,x,x,x
x,-,1,-,x,x
-,x,-,x,-,x
x,-,x,x,x,x
x,x,x,x,-,x
x,x,x,x,x,x
```

part02test01.moves.eps

```
d,d,u,u,u,u,r,u,d,u,l,l,u
```

part02test01.solution

```
game
[ 'x', 'x', 'x', '-', 'x', 'x' ]
[ 'x', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', '-', '1', 'x' ]
[ '-', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', 'x', 'x', 'x' ]
[ 'x', 'x', 'x', 'x', '-', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', 'x' ]
```

4 Many Players , One Move

The third part (`manyPlayersOneMove` in the file `csce322a02part03.js`) will take in one (1) argument (a game) and return a function that takes in one (1) argument (an arry of moves), and returns the game that is the result of each player in the game making exactly one move until each player has completed a move or no player can complete a valid move. Player 1 moves into `1`, Player 2 moves into `2`, 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.

part03test01.game.eps

```
x,x,-,x,x,x,x,x
x,-,x,-,x,-,x,x
x,1,-,x,-,x,-,x
x,x,x,x,x,-,x,x
x,x,x,x,x,x,-,x
x,x,x,x,x,x,2,x
x,x,x,x,x,x,x,x
```

part03test01.moves.eps

```
l,u,l,u,r,r,d,d,u,d,l,r,r,l
```

part03test01.solution

```
game
[ 'x', 'x', '-', 'x', 'x', 'x', 'x', 'x' ]
[ 'x', '-', 'x', '-', 'x', '-', 'x', 'x' ]
[ 'x', '1', '-', 'x', '-', 'x', '-', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', '-', 'x', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', 'x', '-', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', 'x', '2', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', 'x', 'x', 'x' ]
```

5 Many Players , Many Moves

The fourth part (`manyPlayersManyMoves` in the file `csce322a02part04.js`) will take in one (1) argument (a game) and return a function that takes in one (1) argument (an arry 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 no player can make a valid move. Player 1 moves into `1`, Player 2 moves into `2`, etc. (you may assume the highest number in the provided game is the number of players in the game).

part04test01.game.eps

```
x,1,-,x,-,x
x,x,x,x,x,-
x,x,x,x,x,x
x,-,x,-,2,-
-,x,-,x,-,x
x,-,x,x,x,x
-,x,-,x,-,x
x,-,x,-,x,x
-,x,-,x,-,x
x,-,x,-,x,x
x,x,x,x,x,x
```

part04test01.moves.eps

```
d,l,r,l,l,u,d,u,u,u,r
```

part04test01.solution

```
game
[ 'x', 'x', '-', '1', '-', 'x' ]
[ 'x', '-', 'x', '-', '2', '-' ]
[ 'x', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', '-', 'x', '-' ]
[ '-', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', 'x', 'x', 'x' ]
[ '-', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', '-', 'x', 'x' ]
[ '-', 'x', '-', 'x', '-', 'x' ]
[ 'x', '-', 'x', '-', 'x', 'x' ]
[ 'x', 'x', 'x', 'x', 'x', 'x' ]
```

6 Extra Credit (10%)

Games will contain an arbitrary number of players.

7 Naming Conventions

Your files should follow the naming convention of
`csce322a02part01.js`, `csce322a02part02.js`, `csce322a02part03.js`, and `csce322a02part04.js`.

7.1 `helpers.js`

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

8 webgrader Note

Submissions will be tested with `node.js`, not the browser. `cse.unl.edu` is currently running version 8.16.1 of `node`.

9 Point Allocation

Component	Points
`csce322a02part01.js`	
Test Cases	1×20
Total	20
`csce322a02part02.js`	
Test Cases	1×20
Total	20
`csce322a02part03.js`	
Test Cases	1×30
Total	30
`csce322a02part04.js`	
Test Cases	1×30
Total	30
Total	100

10 External Resources

JavaScript Tutorial