# Assignment 03 (Due: November 8, 2019)

CSCE 322

## Contents



## List of Figures



## 1 Instructions

In this assignment, you will be required to write Haskell functions that facilitate the playing of the variation of Slippery Crossings. 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 a maze and moves to be made.

```
(
"uuddrddlrrd",
[
"xxxxxxxxxxxxx",
"x-----x----x-x",
"x-x------x---x",
"xx-----xx----x",
"x-----x-xx-xgx",
"x-x--x---x1x-x",
"x-xx---xxxx--x",
"x-----------x",
"x-x---xxx-x--x",
"x-x-----xx---x",
"x---x--xx----x",
"xx-x--x------x",
"x--x--x-----xx",
"xx--x---xx-xxx",
"xx--xx--x----x",
"x---x--xx--xxx",
"xxxxxxxxxxxxx"
]
)
```

Figure 1: A properly formatted encoding

The tuple contains a list of moves (`up`, `down`, `left`, and `right`) and a maze (where `x` corresponds to a barrier, `g` corresponds to the goal, `-` corresponds to an empty space, and numbers correspond to players).

### 1.2 `csce322a03part(num).hs`

This assignment requires the implementation of four (4) methods: `onePlayerOneSlide` , `onePlayerManySlides` , `manyPlayersOneSlide` , and `manyPlayersManySlides` . Each method will be implemented in its own file (named `csce322a03part(num).hs`, where `(num)` is 01, 02, 03, or 04). The behavior of each function is described below.

#### 1.2.1 `onePlayerOneSlide (csce322a03part01.hs)`

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

1. A player must slide in a given direction until they hit a barrier or they land on the goal.

```
(
"uuddrddlrrd",
[
"xxxxxxxxxxxxx",
"x-----x----x-x",
"x-x------x---x",
"xx-----xx----x",
"x-----x-xx-xgx",
"x-x--x---x1x-x",
"x-xx---xxxx--x",
"x-----------x",
"x-x---xxx-x--x",
"x-x-----xx---x",
"x---x--xx----x",
"xx-x--x------x",
"x--x--x-----xx",
"xx--x---xx-xxx",
"xx--xx--x----x",
"x---x--xx--xxx",
"xxxxxxxxxxxxx"
]
)
```

Figure 2: Game State Before `onePlayerOneSlide`

```
"Result"
"xxxxxxxxxxxxx"
"x-----x---1x-x"
"x-x------x---x"
"xx-----xx----x"
"x-----x-xx-xgx"
"x-x--x---x-x-x"
"x-xx---xxxx--x"
"x-----------x"
"x-x---xxx-x--x"
"x-x-----xx---x"
"x---x--xx----x"
"xx-x--x------x"
"x--x--x-----xx"
"xx--x---xx-xxx"
"xx--xx--x----x"
"x---x--xx--xxx"
"xxxxxxxxxxxxx"
"END"
```

Figure 3: Game State After `onePlayerOneSlide`

**Sample Sequence**

### 1.2.2 `onePlayerManySlides (csce322a03part02.hs)`

`onePlayerManySlides :: [[Char]] -> [Char] -> [[Char]]` This method will take a list of moves to make and a maze and make each move in accordance to the rules for `onePlayerOneSlide` . If the goal is encountered before the last move in the list is made, the player makes none of the remaining moves.

```
(
"uurlduurldurrrlruldr",
[
"xxxxxxxxxxxxxxxxxxx",
"xx-1x---------x-----x",
"x------xx-----------x",
"x------xx---x-x-----x",
"x-x--x-xx-----x-x--xx",
"x-----x-xx----xx---xx",
"x-x--xx----x--xx----x",
"x-x-----x--x--xxxxx-x",
"x---xx-x-gx--xx-x--xx",
"x-x-xx------x---x-x-x",
"x--xxx-x-xx---------x",
"xx----x--------xx---x",
"x-----x----x-x--x-x-x",
"x-----x----x--x-----x",
"x--xx----x--x--x--x-x",
"x--------x----------x",
"xxxxxxxxxxxxxxxxxxxx"
]
)
```

Figure 4: Game State Before `onePlayerManySlides`

```
"Result"
"xxxxxxxxxxxxxxxxxxx"
"xx--x---------x-----x"
"x------xx-----------x"
"x-----1xx---x-x-----x"
"x-x--x-xx-----x-x--xx"
"x-----x-xx----xx---xx"
"x-x--xx----x--xx----x"
"x-x-----x--x--xxxxx-x"
"x---xx-x-gx--xx-x--xx"
"x-x-xx------x---x-x-x"
"x--xxx-x-xx---------x"
"xx----x--------xx---x"
"x-----x----x-x--x-x-x"
"x-----x----x--x-----x"
"x--xx----x--x--x--x-x"
"x--------x----------x"
"xxxxxxxxxxxxxxxxxxxx"
"END"
```

Figure 5: Game State After `onePlayerManySlides`

**Sample Sequence**

#### 1.2.3 `manyPlayersOneSlide (csce322a03part03.hs)`

`manyPlayersManySlidesHelper :: [[Char]] -> [Char] -> [Int] -> [[Char]]` This method will take a list of moves to make and a maze and make the first move for Player 1, the second move for Player 2... in accordance to the rules for `onePlayerOneSlide` until each player has made a single move each. If the goal is encountered before every player makes one move, the players make none of the remaining moves.

```
(
"dlrruururlr",
[
"xxxxxxxxxxxxxx",
"xx------xx-x--x",
"xxx----xg---x-x",
"xx----xx-x----x",
"x-x-----xx----x",
"x---x-x----x--x",
"x-x----x------x",
"xx----x-----x-x",
"x-----xx---x-xx",
"x-x-xxx-2-x---x",
"x--xxxx4x-----x",
"x-xx----------x",
"x----x-------xx",
"x-x--xx-------x",
"x--xxx--3x--xxx",
"x-x----1------x",
"xxxxxxxxxxxxxx"
]
)
```

Figure 6: Game State Before `manyPlayersOneSlide`

```
"Result"
"xxxxxxxxxxxxxx"
"xx------xx-x--x"
"xxx----xg---x-x"
"xx----xx-x----x"
"x-x-----xx----x"
"x---x-x----x--x"
"x-x----x------x"
"xx----x-----x-x"
"x-----xx---x-xx"
"x-x-xxx2--x---x"
"x--xxxx4x-----x"
"x-xx----------x"
"x----x-------xx"
"x-x--xx-------x"
"x--xxx--3x--xxx"
"x-x----1------x"
"xxxxxxxxxxxxxx"
"END"
```

Figure 7: Game State After `manyPlayersOneSlide`

**Sample Sequence**

### 1.2.4 `manyPlayersManySlides (csce322a03part04.hs)`

`manyPlayersManySlidesHelper :: [[Char]] -> [Char] -> [Int] -> [[Char]]` This method will take a list of moves to make and a maze and make the first move for Player 1, the second move for Player 2... in accordance to the rules for `onePlayerOneSlide` . If the goal is encountered before the last move in the list is made, the players make none of the remaining moves.

```
(
"ulludllrrlddldud",
[
"xxxxxxxxxxxxxxxxxx",
"x------x---x--x----x",
"x---xg-xx---x----x-x",
"x--------xx--------x",
"xx----------x--xx--x",
"x---x--x-x--------xx",
"xx--x-----x-xx--x--x",
"x----------x----x-xx",
"xxx--xxx--xxx------x",
"x--x---------xxxxx3x",
"x-------x-------x--x",
"x-x--x--x-xx---x--xx",
"x2xx----x--------x-x",
"x-1----xx-----x----x",
"xxxxxxxxxxxxxxxxxx"
]
)
```

Figure 8: Game State Before `manyPlayersManySlides`

```
"Result"
"xxxxxxxxxxxxxxxxxx"
"x------x---x--x----x"
"x---xg-xx---x----x-x"
"x--------xx--------x"
"xx----------x--xx--x"
"x---x--x-x--------xx"
"xx--x-----x-xx--x--x"
"x----------x----x-xx"
"xxx--xxx--xxx-----3x"
"x--x---------xxxxx-x"
"x-------x-------x--x"
"x-x--x--x-xx---x--xx"
"x-xx----x--------x-x"
"x21----xx-----x----x"
"xxxxxxxxxxxxxxxxxx"
"END"
```

Figure 9: Game State After `manyPlayersManySlides`

**Sample Sequence**

## 2 Naming Conventions

You will be submitting at least 4 `.hs` files (`csce322a03part01.hs`, `csce322a03part02.hs`, `csce322a03part03.hs`, and `csce322a03part04.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 8.0.2 of `ghc`.

## 4 Point Allocation

| Component | Points |
|---|---|
| `csce322a03part01.hs` | 30% |
| `csce322a03part02.hs` | 20% |
| `csce322a03part03.hs` | 20% |
| `csce322a03part04.hs` | 30% |
| 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