Assignment 02 (Due: Saturday, November 16, 2013)

CSCE 322

Contents

List of Figures

1 Instructions

In this assignment, you will be required to write Haskell functions that facilitate the playing of *Pac-Man* . 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

A example of a properly formatted file is shown in Figure 1. The tuple represents the level, score, lives, map and move list.

```
(154, 3910, 3,
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWWGWWFWB",
"BFFWGGGWFFB",
"BWFWWWWWFWB",
"BFFF_M_FFFB",
"BFWFF_FFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"],
"LUUURUR")
```

Figure 1: A properly formatted Pac-Man encoding

In addition to the map symbols from Assignment 01, there are 2 new symbols: `6` represents a cell that, at the same time, contains a ghost and food and `R` represents a cell that, at the same time, contains a ghost and a power pellet. The symbols in the move list correspond to `Up`, `Down`, `Left`, and `Right`. The input file is correct, so error checking is not required.

1.2 `csce322a2p(num).hs`

This assignment requires the implementation of 5 methods: `trymove` , `replace` , `restart` , `makemove` , and `moveall` . Each method will be implemented in its own file (named `csce322a2p(num).hs`, where `(num)` is 1, 2, 3, 4, or 5). The behavior of each function is described below.

1.2.1 `trymove (csce322a2p1.hs)`

`trymove :: (Int, Int, Int, [[Char]], Char) -> (Int, Int, Int, [[Char]])` This method will take a level, score, lives, map, and move to make, and return a level, score, lives, and map according to the following rules:

1. If the move to make will cause *Pac-Man* to collide with a wall or border, the move is not made and the map is not altered
2. If the move to make will cause *Pac-Man* to enter a cell occupied by a ghost (or ghost and food, or ghost and power pellet), the number of lives that *Pac-Man* has is reduced by 1. The score and map are not altered.
3. If the move to make will cause *Pac-Man* to enter a cell occupied only by food, move *Pac-Man* into that cell (leaving behind an empty cell) and increase the score by 1.
4. If the move to make will cause *Pac-Man* to enter a cell occupied only by a power pellet, move *Pac-Man* into that cell (leaving behind an empty cell) and increase the score by 10.

```
(154, 3910, 3,
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWWGWWFWB",
"BFFWGGGWFFB",
"BWFWWWWWFWB",
"BFFFM__FFFB",
"BFWFF_FFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"],
"LR")
```

Figure 2: Game State Before `trymove`

```
(154, 3911, 3,
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWWGWWFWB",
"BFFWGGGWFFB",
"BWFWWWWWFWB",
"BFFM___FFFB",
"BFWFF_FFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"],
"R")
```

Figure 3: Game State After `try`

Sample Sequence

1.2.2 `replace (csce322a2p2.hs)`

`replace :: [[Char]] -> [[Char]]` This method will place *Pac-Man* and the ghosts into default positions in the map. It is assumed that maps will have an odd number of rows and columns that is greater than 7. The default position for *Pac-Man* is in the center column, 2 rows beneath the center. 3 ghosts will be placed on the 3 center columns of the center row, and the fourth ghost will be placed in the center column, in the row above the 3 other ghosts (See Figure 6).

```
 G
GGG
```

Figure 4: Ghost Alignment After `replace`

Cells that previous contained *Pac-Man* will be set to empty, cells that previously contained a ghost and food will contain food, and cells that previously contained a ghost and a power pellet will contain a power pellet.

```
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWW_WWFWB",
"BFFW___WFFB",
"BWFWWWWWFWB",
"BFFFGGGFFFB",
"BFWFGMFFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"]
```

Figure 5: Game State Before `replace`

```
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWWGWWFWB",
"BFFWGGGWFFB",
"BWFWWWWWFWB",
"BFF__M_FFFB",
"BFWFF_FFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"]
```

Figure 6: Game State After `replace`

Sample Sequence

1.2.3 `restart (csce322a2p3.hs)`

`restart :: [[Char]] -> [[Char]]` This method will alter the positions of *Pac-Man* and the ghosts in the same way that `replace` will. However, `restart` will also store food cells and power pellet cells that were consumed. Empty cells in the corners of the map will be replaced by power pellet cells, while all other empty cells (that won't contain *Pac-Man* or a ghost when the function returns) will become food cells.

```
["BBBBBBBBBBB",
"B_FFWWWFF_B",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWW_WWFWB",
"BFFW___WFFB",
"BWFWWWWWFWB",
"BFFFGGGFFFB",
"BFWFGMFFWFB",
"B_FFWWWFF_B",
"BBBBBBBBBBB"]
```

Figure 7: Game State Before `restart`

```
["BBBBBBBBBBB",
"BPFFWWWFFPB",
"BFWFFFFFWFB",
"BFFFFFFFFFB",
"BWFWWGWWFWB",
"BFFWGGGWFFB",
"BWFWWWWWFWB",
"BFFFFMFFFFB",
"BFWFFFFFWFB",
"BPFFWWWFFPB",
"BBBBBBBBBBB"]
```

Figure 8: Game State After `restart`

Sample Sequence

1.2.4 `makemove (csce322a2p4.hs)`

`makemove :: (Int, Int, Int, [[Char]], Char) -> (Int, Int, Int, [[Char]])` This method will combine the functionality of `trymove` , `replace` , and `restart` to move *Pac-Man* according to the supplied move list. Starting with the first move in the move list, *Pac-Man* will **try** to make each move in the list. If a move results in *Pac-Man* colliding with a ghost, the map should be **reset** and *Pac-Man* should **try** to make the *remaining* moves in the move list. If *Pac-Man* consumes all of the food and power pellets, *Pac-Man* should **restart** the map at the next highest level and **try** to make the *remaining* moves in the move list.

1.2.5 `moveall (csce322a2p5.hs)`

`moveall :: (Int, Int, Int, [[Char]], Char) -> (Int, Int, Int, [[Char]])` After *Pac-Man* **makes** a move, the ghosts must move. A ghost should attempt to minimized the Euclidean distance between itself and *Pac-Man* . If multiple valid moves will achieve this, the ghost should prioritize its possible moves by the order: Up, Right, Down, and Left. Ghosts cannot occupy the same cell on the board. Ghosts are updated from left to right and from top to bottom. If a ghost is blocked

by another ghost on its right or below it, that ghost will be unable to move until the next update, even if (later in the current update) the ghost blocking its path moves. It is possible for a ghost to become trapped in a position where a wall lies between it and *Pac-Man* . In this case, there is no need for alternate logic. The ghost will simply attempt (and fail) to move in a way that minimizes the distance between it and *Pac-Man* until *Pac-Man* moves into a position where the ghost is not obstructed by the wall.

`moveall` should move *Pac-Man* , then the ghosts.

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 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 one `.zip` file that contains `csce322a2p1.hs`, `csce322a2p2.hs`, `csce322a2p3.hs`, `csce322a2p4.hs`, `csce322a2p5.hs`, and `README.txt`. The name of this file should be `csce322assignment2.zip`.

3 Point Allocation

Component	Points
`csce322a2p1.hs`	25
`csce322a2p2.hs`	15
`csce322a2p3.hs`	15
`csce322a2p4.hs`	25
`csce322a2p5.hs`	15
`README.txt`	5
Total	100

4 External Resources

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