ASSIGNMENT 2

Name: Tay Shao An
UID: u7553225
Laboratory time: Wednesday 5:00pm
Tutor: Pramo Samarasinghe

INTRODUCTION

Assignment2 is a farm simulation program that allows the user to design and visualize the total income of a farm across a specified number of years.

PROGRAM DESIGN

FARM.HS - In charge of the calculations within the simulation, including the calculating of degradation rates of each paddock and the total income of the farm.

> Datatypes:
- Farm - stores a list of all the paddocks and a width and height
- Paddock - stores a crop, its yield, and degradation
- Crop
- Location

> allocatePaddock - Takes a crop and assigns it its respective Paddock.

> cycleCrop - Takes a crop and returns the next crop in a list. Mainly used when the user clicks on squares in the simulation.

> renderPaddock - Function for displaying Paddocks on the screen using codeworld's Picture datatype.

> updateFarm - Takes one "step" in the simulation, calculating a new yield and degradation rate for all paddocks in a farm.

> avgNeighbours - Obtains the average degradation rate of all neighbours obtained from the function allNeighbours.

> allNeighbours - Produces a list of the degradation rates of all the neighbouring Paddocks surrounding a Paddock.

> get - Returns a Paddock located at a particular Lot in a farm.

> allLocations - Takes a width and height and returns a list of all possible locations.

> predictIncome - Predicts the income of a farm across a number of years.

> toCrop (in TestPatterns.hs) - Converts a character into a crop

ASSUMPTIONS

> The simple colour scheme of red green and blue was used to represent each of the crops for clarity's sake.
> For renderPaddock, the yield is divided by an arbitrarily big number, 30000, in order for the user to better notice the variation in the colour of the paddock.

TESTING

> Simple functions like cycleCrop, allocatePaddock and allLocations can be tested by running cabal v2-repl comp-1100 assignment2, providing an input and checking the output.
> FarmTest.hs primarily tests for 4 main functions:

- allNeighbour
- allLocations
- get
- predictIncome.

> The functionality of the UI was tested by:

- Clicking on various squares and ensuring that the selected paddock changes.
- Repeatedly “stepping” forward in the UI
- Pressing all available keys and making sure that there is a correct output.
- Ensuring that the correct diagram was generated for each example farm

REFLECTION

> Encountered problems:

1. Initially, I had difficulty implementing the function to calculate the degradation rates of neighbouring paddocks around a specified paddock, as I used a convoluted way of checking for every single possible type of position (e.g. at the left lower corner of the farm, at the right upper corner of the farm…). To fix this, I decided to use the get function, which would simply return a Nothing datatype if no paddock is located in a specified area.
2. The tasks provided seemed easy at first, and I started coding without breaking the functionalities further down. This led to many exceptions occurring and tests repeatedly failing due to small errors which I could not discern. Eventually, I improved my code by utilising smaller helper functions in order to segment functionalities and allow for easier bug testing and readability.

> If I were to redesign the program from scratch, I would:

1. Allow the user to create their own farm by allowing them to input a width and height.
2. Include more crop options to give the software more robustness.
3. Include mathematical formulas to ensure farms such as happyCow do not scale endlessly.
4. Move renderPaddock to GridRenderer