Coursework report:

What the program is and how to use it:

For the project I decided to make a version of the game called which is a game where you guide a constantly moving snake towards a food, usually an apple. You can only the control the direction the snake for every apple it eats the snake increases in length. This in turn makes the game more difficult as if at any point the snake hits the edge of the window, or its head hits another part of its body, the game resets from the beginning,

There is no complex way to play the game all the user does is control the direction of the snake using the arrow keys on the keyboard

The primary library/libraries used:

For the project I used the following libraries:

- Gloss: as this was the most convenient library to display interactive graphics to the user
- Random: so that we could generates the random coordinates for the apple in the game

Any architectural choices made:

First, I created the type SnakeTile which stored a tuple of 2 Integers so that it could be used to store of parts of the Snake as well as apples. Also, I created a second type called Snake which was a list of SnakeTiles. This was as I saw the easiest way to display and move the snake in the game would be a serious of squares, of which the location of each one is stored in a SnakeTile in the list Snake.

I also created two data structures; the first being direction. This is a data type that can either be equal to “UP”, “DOWN”, “LEFT”, or “RIGHT” which is used to store a direction of a snake in the game as it can only move in those four directions. This data also derives (Eq, Ord) so that we can compare directions to each other later in the program. The second data I created was called Grid and used to represent all the important information of the game at any point. Within each grid the following things were stored:

- applePlace (SnakeTile): stored the coordinates of the current apple in the game
- snakePlace (Snake): Stores the list of coordinates for each square that makes up the snake
- facing (Direction): stores the current direction the snake is traveling
- store (Direction): stores the most recently input direction by the user
- expand (Boolean): True if the snake should increase in size
- nextRandom (StdGen): used to generate random numbers for the next coordinates of the apple

For the apple coordinates are much smaller in size as I will multiply the coordinates by 40 later so no two apples could not overlap whereas, the snake needs to run move more smoothly so the coordinates are not multiplied so it can move smaller distances each time to make the movement

seem more natural. To create the structure as well as handle any Inputs and Outputs I used gloss' play function which meant I also had to create all the following data and functions to input into the game functions for the play function to work:

- Just a Display data type that just creates a window for the game to display its outputs /9calle window)
- The colour of the background of the game which is black
- How often pre second to refresh the screen in which case I decided 45
- startGrid which creates the apples first position and creates a list of SnakeTiles to represent the snake
- drawGrid what to output and how for any given Grid in which it just outputs a 40 by 40 square for each snakeTile in each given coordinate as well as a circular red apple at the appropriate coordinate
- handleInput what the program should do for user input which in this case take any arrow direction input and save it in the store variable as long as it is not the opposite direction to which the snake is currently traveling to stop the snake imploding
- update: used to update the state of the grid after each time interview is passed (1/45). This function does a lot as it determines whether the snake should expand in size, the apple should change position or if the current of the direction needs to be updated to what the user input into store as well as handling the movement of the snake

Your personal experience with writing the program:

When I first started making this program I initially struggled to understand how to use the gloss library and looking at an example project of a different function this helped me get the grasp of the library enough so that I could use the play function and the creation of the game became much more straight forward.

I originally wanted the snake to move faster as the game went increasing the difficultly the long the user lasted. However, looking into gloss library and the play function it soon became clear that there was no obvious or clear way to accomplish this.

However, I was able to improve the game in some ways I initially did not attend. As when I first wrote the program and had less confidence in using the gloss library, I wrote the game so that the snake moved a whole block of 40 x/y at a time as if it did not maintain that it could only turn on every 40^{th} x/y, it would no longer align with the apple on the grid which would make the game unplayable. This also had the error that if you pressed to arrow keys quickly enough you could get the snake to go back on itself instantly ending that current game. So after making this original version of the game I then had the confidence to improve the program's complexity, by adding the store variable to the grid data so that the snake's direction would get changed when its x and y coordinate where next at a multiple of 40. This improvement allowed smoother movement of the snake in the game as well as removing a glitch.

To conclude my experiences making this project I feel much more confident using the gloss library now than I did at the start on my project and I would feel more interested in looking at other libraries with monadic interfaces for future projects.

A full list of any resources read or used

https://hackage.haskell.org/package/gloss

https://gist.github.com/gallais/0d61677fe97aa01a12d5

https://hackage.haskell.org/package/random-1.2.1/docs/System-Random.html