For my implementation of pong, I have created the game step by step by first drawing the components of the game onto the canvas, animating the ball using observablesexamples from week 4 as a reference, creating functions for collision detection, updating the scores of the game and lastly adding extended features to the game such as Power Ups. I will be explaining each part of my implementation in detail here.

Firstly, for drawing the components on the canvas. I have created multiple “Const” rectangles using the createElementNS function that is able to create “Rects” and “Ellipses” and from there I was able to make use of the forEach function to set the attributes of them accordingly. For the special icons such as the power ups, I have used an online SVG path tool to create them and have added them to the pong.html page. As a result, I was able to access these attributes using the getElementbyID function and translate them dynamically through my own created function called random_pos which takes in a function called generate_pos that generates a type of vector with x and y position and sets the attribute of the svg element to that position. Therefore I was able to functionally set the position of the elements using my newly created function. Also there would be no side effects that would occur from this as it is a pure function. In order to move the paddle, I had to create a mousePos observable which is referenced from the Observableexamples in week 4. However, I modified the function to take in the paddle of the player and map the y attribute of the paddle to the cursor of the user.

Secondly, for bouncing the ball, I referenced the animatedRect from the ObservableExamples as well in order to change the position of the ball using observable which will continuously add to the x position of the ball using the Map Function. In order to update the states of the game, I have used subscribe throughout my observables. Next, now that the ball is able to move, I need to update the starting ball each time the ball reaches the end of the canvas on either side. Therefore, I have also created another observable that will check if the ball has reached the end of either side of canvas and update the scores of the players as well as setAttribute of the ball back to its original position.

Thirdly, I created multiple Collision functions that will return boolean values so that is easier for me to filter for these conditions and set the attributes of the x and y velocities of the balls in their respective observables. The way I have created the Collision functions is based on several “Or” Conditions that will be filtered off in observable and changing the state of the x,y velocities using subscribe.

Lastly, I have implemented the power ups by creating SVG Path elements using online Path creator. I was able to move the positions of these Power ups randomly using my random_pos() function that also is derived from the generate_pos() function. Therefore, I can pass in the a <Pos> type and the Element as parameters for the function to be invoked and cause the position of the power up to change. I had to create extra Collision functions to check if the SVG Path Element and Ball have collided as the SVG Path element does not have x and y attributes I needed to get it by the BoundingClientRect in built function. When the Lightning Power up is collided with the ball, the ball’s x_velocity gets faster, so it acts as a speed up power up. On the other hand for the “Plus” sign icon, it acts as a power up that will increase the size of the paddle’s height. But i have made it increasingly difficult to encounter this power up as it will make the gameplay easier for the user. Therefore, I created a specific observable which pipes multiple filters together just to watch whether the

Enemy score has reached four and that the ball has collided with the top and bottom wall, in order to make the power up appear in a “random” fashion.

Throughout the assignment, I have tried to follow the functional programming style by implementing Observables for each component that needs to be modified in a reactive way such as Collisions, Updating Scores, Moving the paddle and even for Generating the Positions of the Power Ups. The speed of the ball is also changed through observables. I have made use of filter within my observables to check for certain conditions such as whether the ball has reached a certain position or collided with an element or even if the score has reached a certain limit.

I have managed state in the game by creating Readonly types such as Pos which contains attributes x and y. I can then read the Pos (x,y) to translate the positions of the SVG Path elements. I have also included scoreEnemy, scorePlayer, xvelocity and yvelocity which is always updated throughout the course of the game through observables and by adding subscribe function to these game attributes. Therefore, when the scoreEnemy or scorePlayer reaches 7, all subscriptions will become unsubscribed and the whole game will stop. In conclusion, I keep on filtering for conditions to turn true, and when they do I create subscriptions or unsubscribe as a result of these conditions and this is how I managed the State of the game.