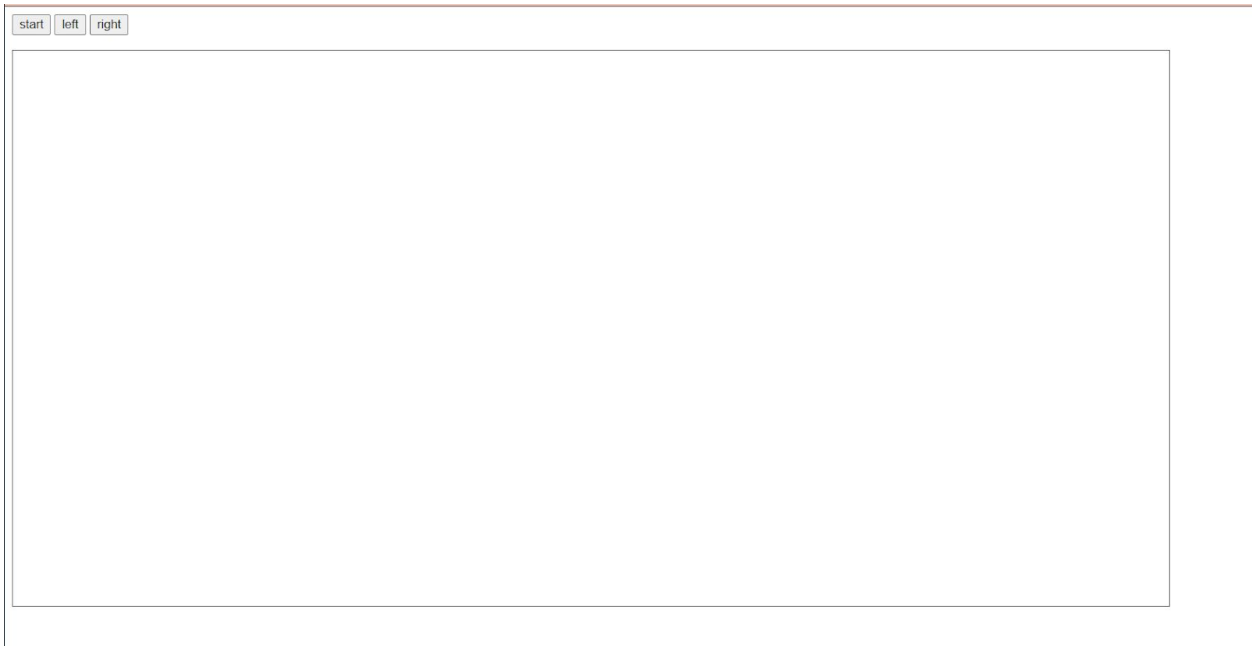
COM S 319 HW3 REPORT

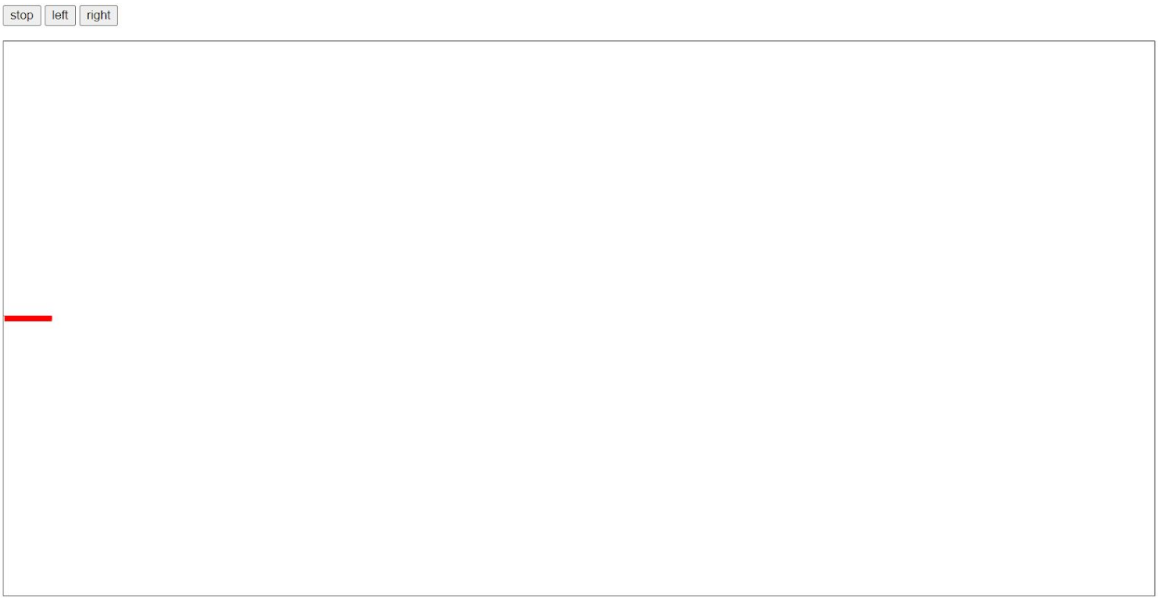
Task 1:

For task one, I decided to give each button an onClick response. The “start/stop” button will start or stop the snake game, and the “left” and “right” button will make the line go left or right respectively.

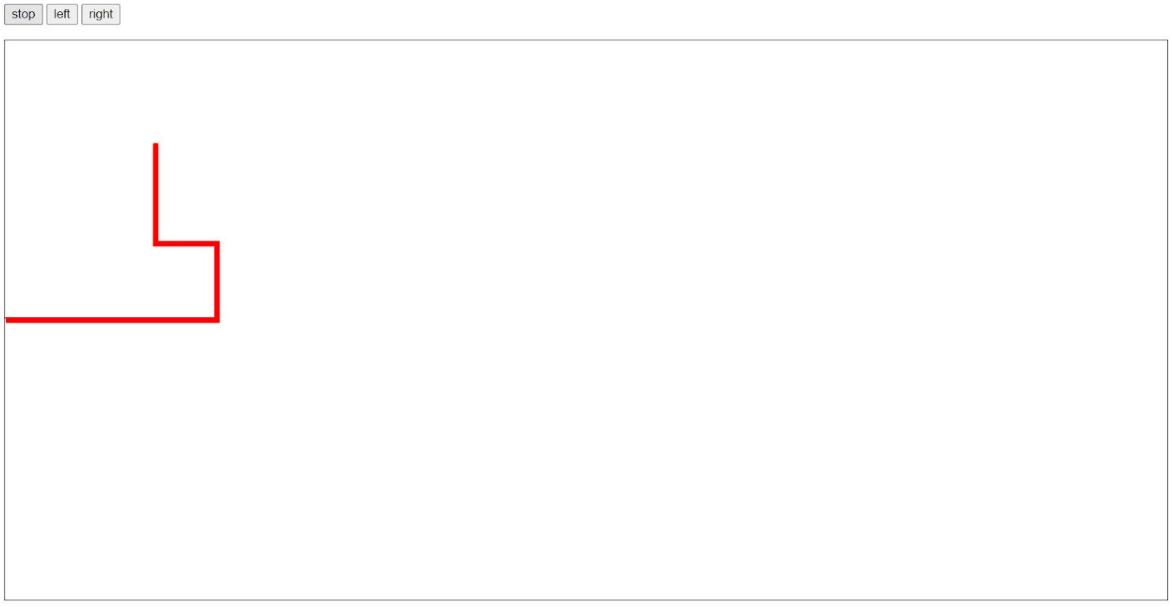
The “start/stop” button calls a function “startStop()” from snake.js. If its value is not “stop”, it will change it to stop and call the function “startGame()” to start the game. Before it calls the function, it checks whether there are any saved positions already in the game. Below is a screenshot of what the user will see when they first load the page.



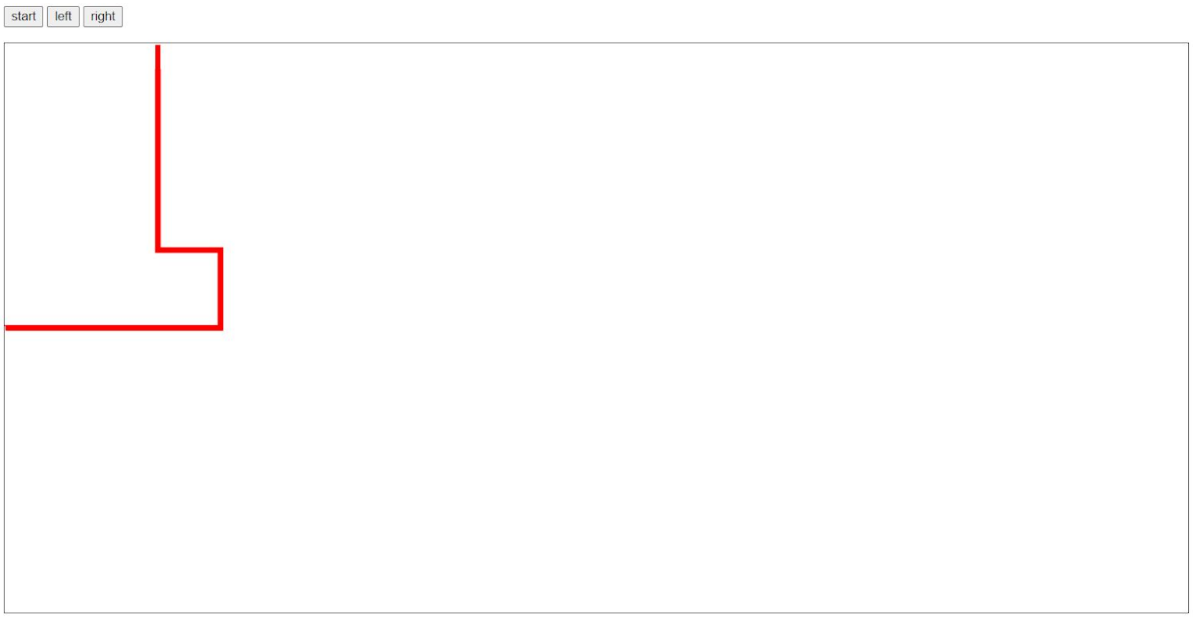
Once the user clicks the “start”’ button, a line will appear from the centre left of the canvas, as shown below.



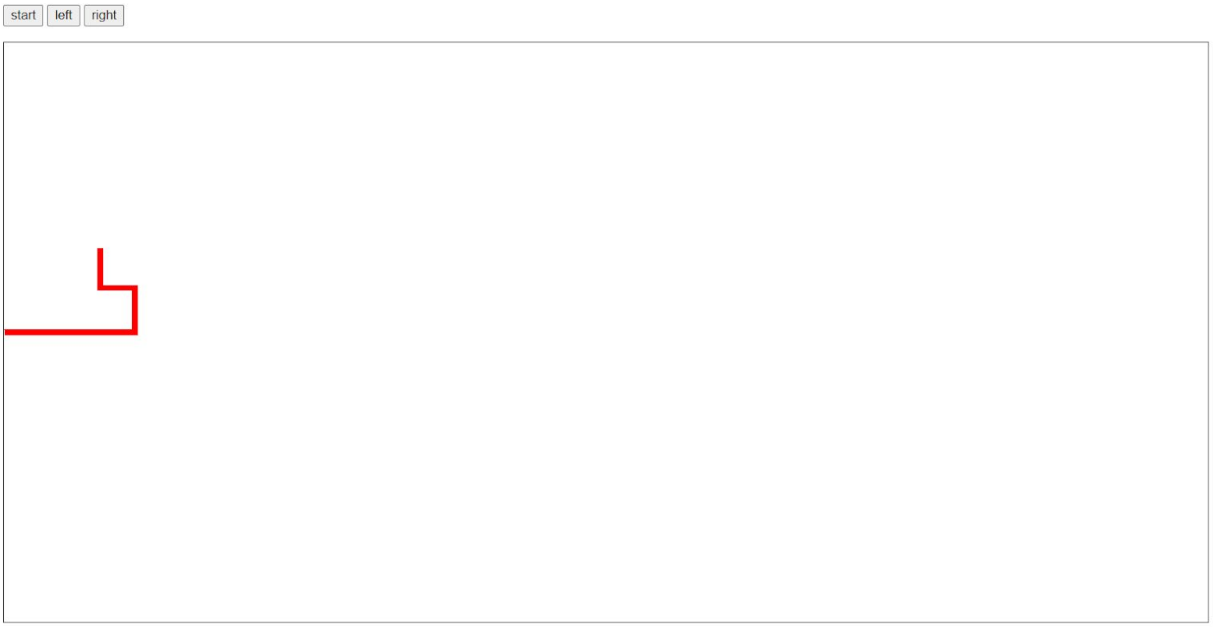
Clicking the “left” or “right” button will make the line go left or right as mentioned above. This is done through the “left()” and “right()” function, which changes which values to increment or decrement accordingly, and is passed back to the “startGame()” function as the timer runs on to update the line. Seen below is a screenshot taken after clicking the left button twice and the right button once.



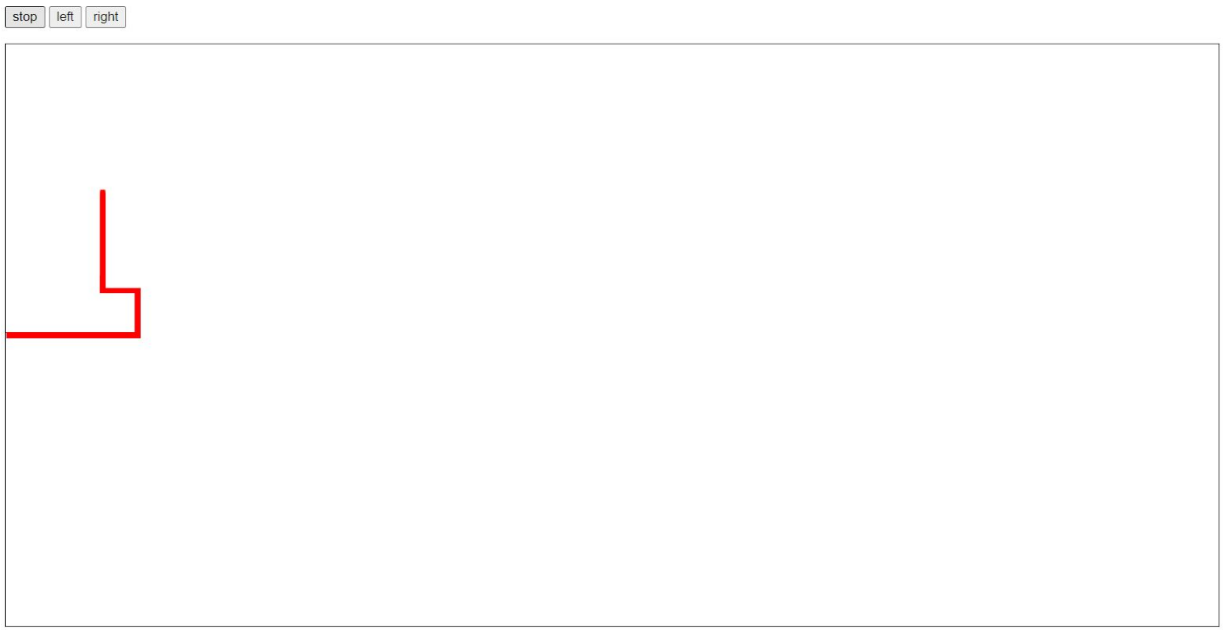
Once the line reaches the edge of the canvas, it will stop the game and the stop button will show a value of “start”.



The user can also choose to pause the game while playing by clicking the “stop” button. This will change it’s value to “start” and the line will stop moving.



By clicking the “start” button again, the line will continue moving from its saved position.



This is done through the “startStop()” function, where the current position of the line will be saved and the movement of the line will be set to zero. The saved position is used when the user clicks on the “start” button again, so that the user can continue where they left off.

I was unable to implement the portion where the line would stop when it hits itself. I was looking into using the getImageData method, but in the end I was unable to implement it the way I wished. The idea was to have a const for a portion of the whitespace, and then constantly compare it with wherever the line was headed to to see if the next spot was red or white. If it was not white, it would mean that the line had already passed through there before and would be unable to continue.

Task 2:

For task 2, my outputs mimic the example shown in the assignment PDF.
Here are some examples of inputs:

```
C:\Users\Evelyn\OneDrive\Documents\COM S 319\hw3>node hw3.js
1st Number: 142
2nd Number: 53
3rd Number: 3123
4th Number: 421
Factorial of the 1st number is =  2.69536413788816l4e+245
The sum of all the digits of the second number is =  8
The Reverse of the 3rd number is =  3213
Is the 4th Number a Palindrome(True/False)?  false
```

```
C:\Users\Evelyn\OneDrive\Documents\COM S 319\hw3>node hw3.js
1st Number: 4
2nd Number: 312312
3rd Number: 42141
4th Number: 1234321
Factorial of the 1st number is =  24
The sum of all the digits of the second number is =  12
The Reverse of the 3rd number is =  14124
Is the 4th Number a Palindrome(True/False)?  true
```

The implementation is described as follows.
For the factorial, I employed the help of a for loop and kept multiplying numbers starting from 1, incrementing by one each iteration of the loop until we reached the input number. (Thus, if the example given was 4, the loop would multiply 1 x 2 x 3 x 4)

For the sum, I used a while loop and incremented a base variable of 0 with the remainder of the input number divided by 10, while there was still value in the input number. I also made sure to reduce the input number with each iteration by dividing it by 10 each time and used Math.floor to ensure that the reduced input number was still an integer instead of a number with a decimal.

For the reverse of the third input number, I converted the input number into a string, and employed the use of the .reverse() method. However, I had to split the string up into individual characters first, reverse them, then attach them back together. This can be achieved using the split(“”) and join(“”) methods.

Finally, for the palindrome checker, I figured that if it was a palindrome, the reverse of the input number would be the same as the input number itself, so I used the same method as I did for the third number, and then checked for true or false on whether the reverse was the same as the input.