JS Lab 4:

Due Web, May 16 at midnight (NOTE THE DATE AND DAY!)

Part 0: Study for the Celebration of Knowledge. Be aware that you do not memorize programming. It is important to be able to understand programming. It's like reading – yes you can memorize the words and the syntax, but the words and syntax can be put together in so many ways that you must understand the text in order to follow anything that is written.

Part A:

Your final project: If you haven't done this, complete the following:

1. Start your javaScript.
   For this project you will need a lot of arrays: Keep in mind that you can always substitute what the array is at a particular location for that thing. So, for instance, if you've got an array of ids and arr[3]='picid3'
   Then in your code you can say,
   document.getElementById(arr[3]).src = "dog.jpg"
   And it's exactly the same thing as saying,
   document.getElementById('picid3').src = "dog.jpg"
   Equally, if you have an array of numbers, so you've got:
   topArr[2] = 100

   then in your code you can say,
   document.getElementById('picid').top = topArr[2]+"px"
   and it's exactly the same as saying,
   document.getElementById('picid').top = "100px"

   a. Create an array of the ids of your bottom monsters (the monster images at the bottom of your web page, each with its own id.
   b. Create an array of random numbers. This array should be exactly the same length as your first array. The random numbers should be initialized to something between 150 and 350 (although you may wish to play with this range later on) These will be the maximum top position your monster will get to.
   c. Create a third array for the Monster's current position from the top (so a third array of integers the same length as your first array, initializing each value in the array to something like 800, although again you may wish to play with that number later.
   d. Create yet another array the same length as the monster array. This will control the direction the monster is moving in. Each value should be initialized to -10 (because we want these monsters to move towards the top).
   e. Final bottom monster array: create one last array the exact same length as the monster array. Each value in the array should be set to the corresponding monster's position from the left on the screen (so look at your css and see how far over you positioned it from the left.)

f. Now repeat the entire thing with a second set of 5 arrays for your monster's at the top. Difference? The random numbers should be closer to between 50 and 250, the top position array should start at maybe 5 (closer to the top), and the direction should be initialized to 10 (so they're going in the opposite direction from your bottom monsters).

g. Create a function. Inside your function create a variable x. Set the variable x to be a number between 0 and the length of your bottom monster array. X will be used as the index into all of the arrays associated with the bottom monsters. So, for instance, if you want to access the monster's id, you'll use the monsteridarray[x] (or whatever you happened to call that array). If you want to access the monster's current position from the top, you'll use the monstercurrtoparr[x] (or whatever you happened to call that array).

You are going to change the position of your monster using the arrays you created above:

- First, add the monster direction array at index x's number to the monster top position array (so it will actually go down by 10).
- Next use document.getElementById and the monster id array to set monster x's position to be absolute
- Then use document.getElementById and the monster id array at x to change the position of your bottom monster to its new top position in the monster position array at index x.
- At the bottom of the function, use setTimeOut to call this function every 100 msec (you can play with this number until you get the speed you want later).
- For testing purposes, make the start button on your web page call this function and test to make sure that one of your monster x moves upward. (we'll change what the start button calls later).
- Add a condition: if the monster x's top position gets to be less than the monster's max top position, change the monster x's direction from -10 to 10.
- Test again, and make sure your monster x goes up and then down.
- Now add another if condition: if the monster x's top position gets to be greater than a particular bottom number (maybe 800, although you can play with this), change the monster x's direction from 10 to -10. In addition, since you want the monster to move up to a different maximum top position each time, change monster x's top position to a new random number between 150 and 350.
- Test again. Now your monster should be moving up and down, each time moving up to a different height. Try changing the value of x and seeing if different monster's move up to different heights.

Part B: Making Hero Move:

In your javaScript you'll need:

2 variables:

Qne variable will hold the number of pixels your hero image will be positioned down from the top, and one variable will hold the number of pixels your hero image will be positioned from the left. Set their

initial values to be the number of pixels down from the top and over from the left where you want your hero to start on your web page.

Add a function to move the hero to the left.

So inside the function, position the hero image absolutely (using getElementById for the image on your web page). And then to move to the left, subtract 10 from the leftpos variable and, again, using getElementById for the image of the hero on the web page, move the image 10 pixels to the left. (This is like the moving car example in class).

Now in the html part of your page, have the left-arrow button call the function you've just written with onClick.

Test this to make sure that when you click on a button, the hero moves to the left. Note that if the hero "jumps" the first time you call it, the very first top and left values that you set your variables to are different from where your hero is positioned on the web page initially. To fix this, change either the css positioning of your hero or the variable initial numbers so that the 2 numbers are the same for both the top and the left positioning.

Add a function to move the hero down.

Again, inside function, position the hero image absolutely. And then to move down, subtract 10 from the toppos variable and, again, using getElementById for the image of the hero on the web page, move the image 10 pixels down.

Now in the html part of your page, have the down-arrow button call the function you've just written with onClick.

Test this to make sure that when you click on a button, the hero moves to the down.

Part C: Parameters:

Now you're going to add a parameter to the function you wrote above, between the two parentheses.

First let's start with the 2 functions that move your hero.

In the function for moving your hero left, add a parameter. The parameter will be used to indicate whether the hero will be moving right or left. If the parameter holds the word 'left', then you should subtract 10 from the left variable and reposition the hero image (exactly like you've already done – the only difference is that you only do it if the parameter holds the word left). If, however, the parameter holds the word 'right', then you should add 10 to the left variable and reposition the hero from the left (remember, we're always positioning in from the left, which is why you either add 10 or subtract 10 from the left variable to move the hero left or right).

Now In your left arrow image, add the word 'left' to the function call, between the parentheses. Check to make sure that when you click on it, the hero still moves 10 pixels to the left.

In your html code, add an image of a right arrow. Inside the image, add a call to the function written in part B, with the word 'right' between the parentheses. Test it to make sure that when you click on the right arrow, the hero moves to the right and when you click on the left arrow, the hero moves to the left.

You will also want your hero to be able to move up as well as down. In the updown function, add a parameter. That parameter will either hold the word "up" or "down". If the parameter holds the word

'up', subtract 10 from the top variable and position the image using getElementById from the top. If the parameter holds the word 'down', add 10 from the top variable and reposition the hero.

Test your arrows to make sure they all move the hero in the appropriate direction.

Now let's make all the monster's move.

You have 2 monster moving functions. If you have done them correctly, they have a variable inside them that holds a number between 0 and the length of all the arrays. Whatever you called that variable, make just the variable be your parameter, and get rid of the variable inside the function. So, for instance, if you currently have something like this (I made up your function name):

```
function MoveBottomMonsters() {

    var x = 3

    …
```

Change this to:

```
function MoveBottomMonsters(x) {

    …
```

For now, don't test this. Do the next step (the start function) below.

Part D: Starting function:

We're going to make this program start in a different way.

Why: we want all the monsters to start moving simultaneously. To do that we must call the 2 different move monster functions simultaneously. Worse, we must call them with every index in the array. So, if you called your first move monster function MoveBottomMonsters, then you'd have to call it as follows:

```
MoveBottomMonsters(0)
MoveBottomMonsters(1)
MoveBottomMonsters(2)
…
```

For every monster in the array.

The problem is, you can't call more than one function with only one button click on your web page.

So instead, you can create a starting function, and place all the different calls to the movebottommonster function in there.

You can also place all the different calls to the movetopmonster function in there as well.

In part A, you had a start button that called the move bottom monster function. You used that to test to make sure the bottom monster moved up and down. Now change the start button so that, instead of calling the move bottom monster function, it calls the starting function.

Test it by clicking on your start button. Hopefully all your monsters start moving up and down if you've done this correctly.

Part E: Adding the score

Adding the bad scoring:

This is actually a modification to the 2 move monster functions (the one for moving the bottom monsters, and the one for moving the top monsters). You're going to add scoring to the monster function.

First, make sure you have a paragraph to your web page. Give the paragraph an id. This is the paragraph that is going to hold the score.

Second, add a score variable to the javaScript script (up at the top where you've got all the other variables. Set it to 0 to start with.

Now, in the monster moving functions (not the start function), add a check to see if the left position of the hero is within 20 pixels of the left position of the monster (e.g., if ((lefthero > leftmonster[x] – 20) && (lefthero < leftmonster[x] + 20))) – note that you may wish to adjust the number 20 if your monster is significantly bigger than 20 pixels in size. This is all very similar to the car running over the frog, but in this case, we also want to check to ***see if the top positions of both the monster and the hero are within 20 pixels or so of each other,*** so you'll have to add a check for that as well.

If the monster is over the hero on the screen, the score should be modified by subtracting 10, and you should use getElementById to change the innerHTML of the paragraph on your web page to the new score.

Test this by clicking start to get the monster moving, then moving the bird so that the monster goes over the bird. Make sure the score changes.

Adding the good scoring:

Currently if the monster is over the hero, the score decreases by 10 points. But if the hero makes it across the board, the score should go up by 10 points. So if the hero's left position is over the edge of the board's edge (whatever edge you've chosen), the score should go up by 10 points. You will need to modify the move hero function so that includes an if condition that will change the score and write it to the paragraph.

Once the hero has reached the other side of the board successfully, on top of the score going up to 10, the hero should be repositioned back to its beginning position at the other side of the board.

In my function, I added one more check: if the score was greater than or equal to 100, I changed the paragraph to say, "You win!!!"

Save and test your code..

That's it! You've got a game!! Make sure you've completed the html and css so that the game looks good and that everything is positioned initially where you want it and htat it doesn't "jump" when you first start the game. Remember, when you position something in the html you are positioning it a certain number of pixels from the left and the top. If these numbers aren't identical to the numbers you're using as starting values in your javaScript, the image will appear to jump to the position you specified in your javaScript.

Make sure when you turn in this, you upload all images as well as your html and css code to the server, and turn in the url to the TA. Again, easiest way is to just upload the folder that holds all your code and images.

Extra Credit 1 (5 pts):

You can modify your monster move functions so that when the bottom monsters get to the bottom, they pause for a random number of seconds before starting again. In other words, normally within the function setTimeout is used to call the function every 100 milliseconds (or whatever relatively fast number you chose). But to modify it so that the monsters pause a random amount before starting to move again, you can generate a random number greater than 100 and less than some max number of milliseconds you might want a monster to wait, and then call setTimeout just that one time (when the bottom monsters get to the bottom, and when the top monsters get to the top) at that random number.

Extra Credit 2 (10 pts):

If you are interested, you can modify the code so that it uses keystrokes to move the hero instead of having to click on the arrows on your page. It makes moving the bird much quicker. I've included a brief set of instructions on how to do this on my web site.