Name _____ Period_____

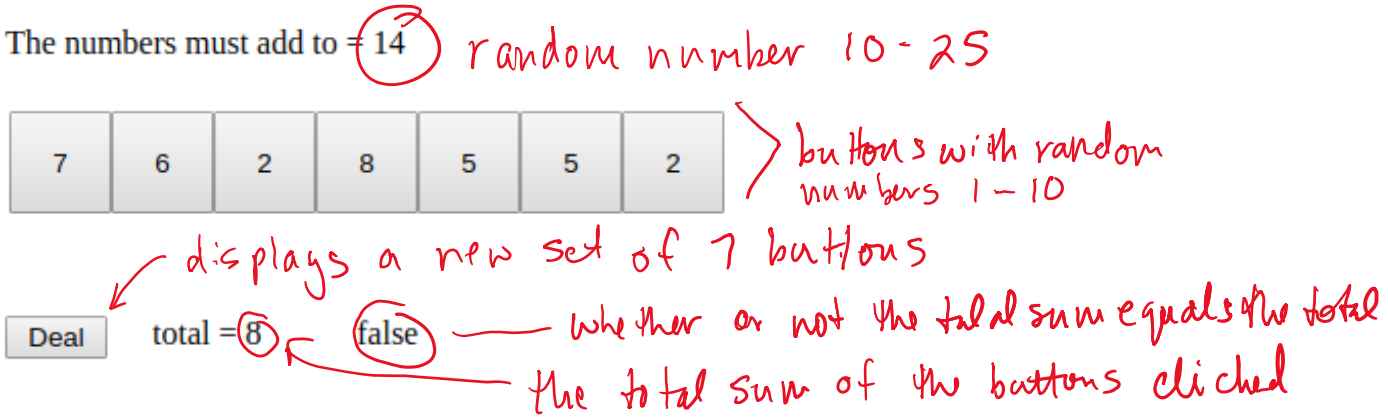
Addition Blocks

Your Tasks (Mark these off as you go)

- ☐ Review the program requirements
- ☐ Write a function to create the buttons
- ☐ Write the sumButtons function
- ☐ Implement the deal function
- ☐ Receive credit for this lab guide

☐ Review the program requirements

The game you will create for this lab is an arithmetic game. How the game works is described below,



When the game loads, 7 buttons appear on the screen. Each button displays a random number 1 thru 10, where 10 is not inclusive.

As the user clicks on the buttons, the *Total* field is updated and displays the total sum of the buttons clicked

The numbers must add to field displays the total sum that the buttons clicked must add to

If the *Total* field is the same as *The numbers must add to* field, the boolean field will display true, otherwise it will display false.

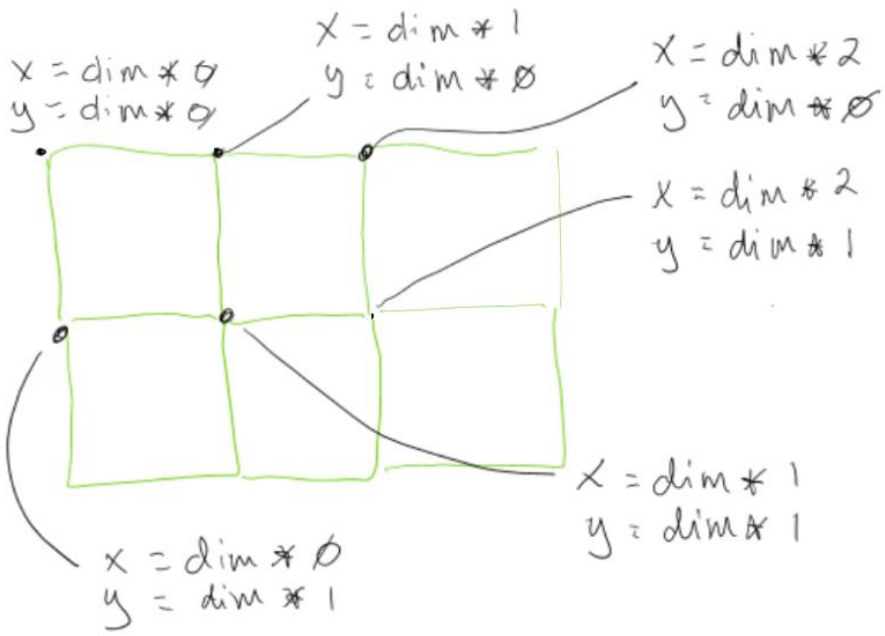
When the Deal button is clicked,

- New numbers are generated for each button
- A new number is generated for the *The numbers must add to* field
- The *Total* field is reset to 0
- The *boolean* field is set to false

☐ Write a function to create the buttons

In the previous labs, you wrote code to create grids of buttons. Below we will review how we did this.

First, we created a function that accepted 3 parameters which represented the dimension, x position, and y position of each button. We then declared a variable (`dim`) to represent the dimension of each button. This new variable allowed us to create many buttons in terms of this dimension. The placement of each button in terms of the dim is illustrated below,



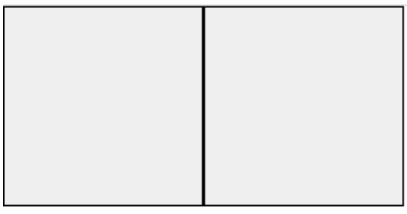
To make our grid interactive required that we add action listeners to each of the buttons. To tell us which button is clicked, we also added an id to help us identify which button was clicked.

The example below illustrates how to create two interactive buttons using our function. `sumButtons` is the function called each time a button is clicked. You will implement `sumButtons` later.

Code	Output

```
var dim = 100;
var b0 = makeButton(dim, dim*0, dim*0, 0);
var b1 = makeButton(dim, dim*1, dim*0, 1);

function makeButton(d, xPos, yPos, id){
    var b = document.createElement("button");
    b.style.border = "black solid thin";
    b.style.width = d+"px";
    b.style.height = d+"px";
    b.style.position = "absolute";
    b.style.left = xPos+"px";
    b.style.top = yPos+"px";
    b.style.textAlign = "center";
```



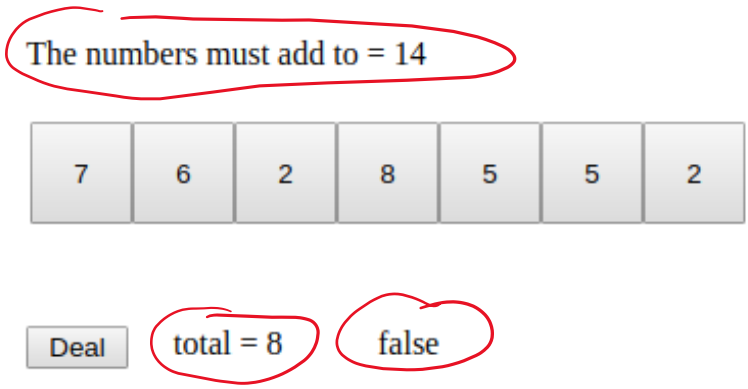
```
    b.style.fontSize = "2em";
    b.id = id;
    b.addEventListener("click", sumButtons);
    document.body.append(b);
    return b;
}
```

Refer to the *makeButton* function above. Write code that calls the function to create the grid shown below. The position of each button should be defined in terms of *dim*.

Code	Output
	6 4 6 2 5 2 7

☐ Write the sumButtons function

For our game to work requires we have three fields. These are circled below.



Each time a button is clicked, the *Total* field must update. The div which holds the total value is defined below,

```
var totalProgressDiv = document.createElement("div");
var total = 0;
```

In the space below, write a function called `sumButtons` that could be used to get the value of the button clicked and update the `totalProgressDiv` with the current `total`.

The following can be used to retrieve the random number from the button.

```
Number(event.target.innerHTML);
```

Once you retrieve the value, add this value to the total and display the new total in the `totalProgressDiv`

```
totalProgressDiv.innerHTML = "Total = " + total;
```

Once the button is used, it cannot be clicked again. In the body of your function, write code that removes the number from the button. To remove the button from just set the innerHTML to empty quotes, `innerHTML = ""`

Now that we know the total of the buttons clicked, we need to update whether or not the total is the same as the random number you created above. If the `total` and `theRandomNumber` variables are the same, the following expression will return true, otherwise, it will return false.

```
(total == theRandomNumber)
```

The boolean field which stores the result of this comparison is defined below.

```
var resultBox = document.createElement("div");
```

Write code that could be used to update the result box with true or false, depending on whether not the buttons add up to the total. This line of code will go at the bottom of your `sumButtons` function you wrote above.

☐ Implement the deal function

Each time the deal function is called the following should occur,

- New numbers are generated for each button
- A new number is generated for the *The numbers must add to* field
- The *Total* field is reset to 0
- The *boolean* field is set to false

When the page loads the deal function is called. The deal function is also called when the deal button is clicked. Consider the following code which could be used to create the Deal button,

```
var dealButton = document.createElement("button");
dealButton.innerHTML = "Deal";
dealButton.addEventListener("click", deal);
```

Complete the `deal` function. In the body of the deal function you must,

- Create a new random number (1 thru 10 (not inclusive)) for each button and display it
- Create a new value for `theRandomNumber` (10 thru 25 (not inclusive)) and display it in the `numsMustAddToDiv`
- Reset the total to 0 and display it the `totalProgressDiv`
- Set the innerHTML of the `resultBox` to false

☐ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.