Name ______________________________ Period _____ Role (Circle one) Programmer/Driver

Name ______________________________ Period _____ Role (Circle one) Programmer/Driver

Boolean Expressions

Your Tasks (Mark these off as you go)

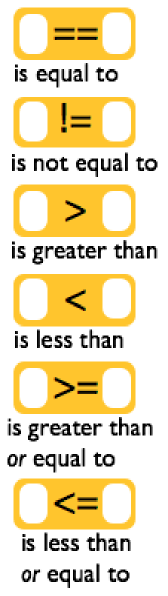
- ☐ Write a Boolean expression
- ☐ Apply the AND and OR operators
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Group Boolean expression
- ☐ Write functions that return Boolean values
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Brainstorm a program
- ☐ Receive credit for the group portion of this lab

☐ Write a boolean expression

- A *Boolean value* is simply a computer science-y term that means a true/false value.
- A *Boolean expression* is a statement that evaluates to a *Boolean value* (a single true/false).

To determine whether two values are the same or not the same, or whether one value is greater or less than another value requires a *comparison operator*.

Below is a list of comparison operators commonly used in computer science.

 == is equal to != is not equal to > is greater than < is less than >= is greater than or equal to <= is less than or equal to	To the left are 6 common comparison operators. Each compares a value on the left with a value on the right and returns a Boolean value – true or false. Most of these do what you would expect. Why these symbols: ==, !=, <=, and >=? 1. We use `==` because the single equal sign `=` is the assignment operator. We need something different to indicate we want to compare two values instead of assign one to the other. Common mistake: writing something like `if (age = 18)` instead of `if (age == 18)`. We'll make sure we get this down later. 2. We use `!=`, `<=`, and `>=` because they only require ASCII symbols. Historically the mathematical symbols `≠`, `≤` and `≥` were hard or impossible to produce on some computer systems. The `!` is universally read as "not".

Refer to the following variable declarations, then indicate the output for each `console.log()`

```
var a = 8;
var b = 9;
var c = a;
var d = “hello”;
var e = “goodbye”;
```

`console.log(a == b);`	
`console.log(a > b);`	
`console.log(a < b);`	
`console.log(d == e);`	
`console.log(d > e);`	

☐ Apply the AND and OR operators

Consider the following example. Suppose that we know that x = 3 and y = 97. Below are statements about x and y and whether or not they are true or false individually, AND whether the entire statement is true or false.

Statement	**True or False**
`(( x < 10 ) AND ( y = 97 )`	First part is true, second part is true, entire statement is true
`(( x < 10 ) AND ( y = -3 ))`	First part is true, second part is false, entire statement is false
`(( x < 10 ) AND ( y ≠-3 ))`	First part is true, second part is true, entire statement is true
`(( x < 10 ) OR ( y = 97 ))`	Either part is true, entire statement is true
`(( x < 10 ) OR ( y = -3 ))`	Either part is true, entire statement is true

Evaluate whether each of the following is true or false for the conditions below,
`x = 11 and y = 5`

`(( x < 10 ) AND ( y = 6 )`	
`(( x < 10 ) AND ( y = 5 )`	
`(( x > 10 ) AND ( y ≠-3 )`	
`(( x < 10 ) OR ( y = 5 ))`	
`(( x > 10 ) OR ( y = 5 ))`	

The above examples illustrate how true and false statements are evaluated, however the syntax is not correct. In order for java to correctly read the syntax the “AND” and “OR” statements must be replaced with the correct symbols as shown below,

Statement	**True or False**
((x < 10) && (y = = 97)	And is replaced with &&, the = is replaced with “= =”
((x < 10) && (y = = -3)	And is replaced with &&, the = is replaced with “= =”
((x < 10) && (y != -3)	And is replaced with &&, the ≠ is replaced with “!=”
((x < 10) || (y = = 97))	Or is replaced with ||, the = is replaced with “= =”
((x < 10) || (y = = -3))	Or is replaced with ||, the = is replaced with “= =”

Refer to the following variable declarations, then (a) Re-write each statement using proper javascript syntax and (b) indicate whether the statement evaluates to *true* or *false*

```
var x = 79;
var y = 46;
var z = -3;
var w = 13.89;
var y = 40.0;
var t = true;
var f = false;
```

Statement	**Proper javascript syntax**	**T/F**
`(( x < 10 ) AND ( y = 46 )`		
`(( x > 10 ) AND ( y = 46 )`		
`(( x > 10 ) AND ( z ≠ -3 )`		
`(( x > 10 ) OR ( y = 5 ))`		
`true AND false`		
`true AND !false`		
`!t OR f`		
`x ≠ 3 OR f`		
`y/2 > w AND w ≠ x`		

☐ Have Ms. Pluska check off the above tasks

Before you continue have Ms. Pluska check off the above tasks

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature ____________

☐ Group Boolean expressions

Just like math operations follow an order of precedence, so too do operations like AND (&&) and OR (||). Consider a problem like,

console.log(false && true || true);

Which part do we do first? As it turns out we would first do && and then ||. Because false and true are not the same thing, false && true evaluates to false. Next we consider false or true, which is true. The order of precedence for the operators we are studying are as follows,

!   ==   !=   && ||

To help avoid the confusion of the order of execution of Boolean statements, you can use parentheses like below,

console.log((true && false) || ((true && false) || false));

Refer to the following code to evaluate what is printed.

```
      var x = 79, y = 46, z = -3;
      var d = 13.89, jj = 40.0;
      var b = true, c = false;
```

`console.log( b && c || !c);`	
`console.log( x == y && !(z < 0) || b && c);`	
`console.log( x != y && y==z && b || !c);`	
`console.log( x > y || c || b && jj%4 != 0);`	

☐ Write functions that return Boolean values

In the previous lesson we wrote functions that returned numeric and text (also called String) values. For example,

```
function rectangleArea(width, height){
     var area = width * height;
     return area;
}

console.log(rectangleArea(5, 7); //prints 35
```

We can also write functions that return Boolean values

```
function rectangleArea(width, height){
     var area = width * height;

     return (area > 100);
}

var myArea = rectanglearea(10, 5);

console.log(myArea);//prints false
```

(a) Declare a function monitorCount() that has two parameters. The first parameter represents rows and the second parameter represents columns.
(b) Let's compute the number of monitors by multiplying rows and columns and then returning the value. In the function body of the function you just wrote, use the return keyword to return rows * columns.
(c) Declare two new variables, budget and monitorCost. Now write a new function called checkBudget. Check budget will call monitorCount which will return the total number of monitors required for our office. To calculate the total cost of all monitors in a 10x5 office you could use, `monitorCount(10, 5) * monitorCost;` Complete the checkBudget method, so that it returns true if the total cost of the monitors is less than the budget or false if it is greater. DO NOT USE if statements

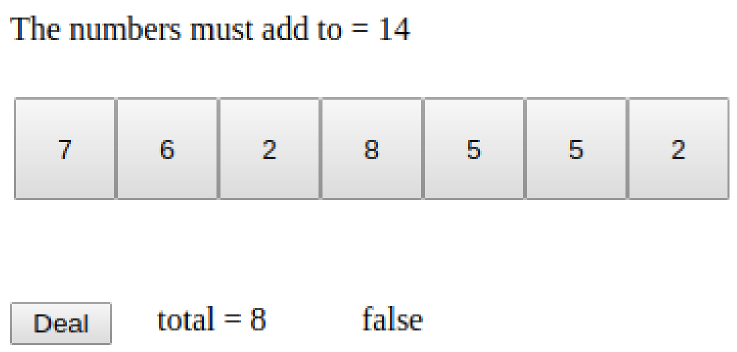
☐ Have Ms. Pluska check off the above tasks

Before you continue have Ms. Pluska check off grouping Boolean expressions and writing fucntions that return Boolean values

Do not continue until you have Ms. Pluska's (or her designated TA's) signature ____________

☐ Brainstorm a program

Below is a screen shot of the game you will create,


The numbers must add to = 14

total = 8 false

A description of the game is below,

- The *number must add to = 14*. 14 represents a random number from 5 (inclusive) to 20 (inclusive)
- The seven buttons. The text for each button is a random number from 0 (inclusive) thru 9 (inclusive)

- The *Deal* button. Each time the deal button is clicked, The value of *The numbers must add to* updates, and the value that displays on each of the seven buttons is updated.
- *total = 8*. The total is initially 0, but this value is updated each time a button is clicked. Once the *total* equals the *The numbers must add to* field, false switchs to true

On a separate sheet of paper brainstorm how you will create this program. Your plan should be thorough and sequential. What will you do first, second, etc. For example,

"Step 1: We will create a variable to hold the "numbers to add to" variable. When the page loads we will create a random number from 5 to 15 and assign the value to this variable.

Step 2: We will create a function that creates 7 buttons. For each button we will also create a random number and assign this to the innerHTML

etc…"

Tips on how to write the actual program will be provided, but you MUST have a thorough plan before you begin.

☐ Receive Credit for the group portion of this lab

STOP

- Indicate the names of all group members.
- Make sure both you and your partner have completed the above tasks
- Have Ms. Pluska check off the group tasks
- Submit your lab to the needs to be graded folder to receive credit for the group portion of this lab.
- Do not submit your lab until you have Ms. Pluska's (or her designated TA's) signature

  ____