Name ____Period ______

While Loops

Your Tasks (Mark these off as you go)

- ☐ Interpret a *while* loop
- ☐ Write a while loop to print a word backwards
- ☐ Write a *while* loop to print a number backwards
- ☐ Write a *while* loop to count the number of heads in a given number of coin flips
- ☐ Write nested *while* loops to create shapes on a grid
- ☐ Receive credit for this lab guide

☐ Interpret a *while* loop

As we learned previously, one of the most important control structures in JavaScript is the *while* loop. A *while* loop is a block of code that is repeated until a condition is no longer true.

Consider a loop that sums all the numbers in a given range. To do this we need a starting point and an ending point. We also need to know how to increment each step along the way.

- start = 0
- incrementing = +1
- stop = 100

The problem of summing all the number in a given range can be solved with the following loop.

```
var sum = 0;
var stop = 100;
var start = 0;

while(start < stop){

     sum += start;
     start++;
}

console.log(sum);
```

Write a loop that prints all the even numbers in a given range.

When writing loops keep in mind the following

1. Initializing expression. The initializing expression can be any integer.

2. Control expression. The control expression indicates how long to continue looping. This is a Boolean expression. As long as the expression is true, the loop will continue.

   Warning: There is something really bad that can happen here. You must write your code so as to ensure that this control statement will eventually become false, thus causing the loop to terminate. Otherwise you will have an endless loop which will crash your program.

3. Step expression. The step expression tells us how our variable should change each time through the loop. In this case we are incrementing j each time. However, other possibilities could include,

   n - -   n = n + 4   n = n * 3   etc.

Indicate the output for each of the following,	

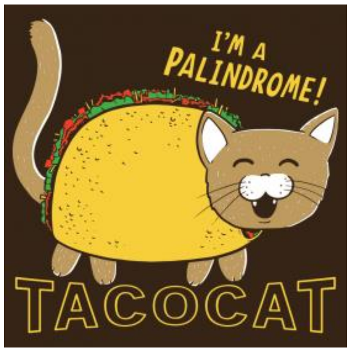
`var j = 0, g = 1;`
 	
`while(g < 10){`
`     j++;`
`     g++;`
`}`
 	
`console.log(j);`	

`var s = 1, j = 4;`
 	
`while(j >= 0){`
`     s = s + j;`
`     j--;`
`}`
 	
`console.log(s);`	

`var j = 0; i = 10;`
 	
`while(i > 0){`
`     i *= -3;`
`     i--;`
`}`
 	
`console.log(i)`	

☐ Write a *while* loop to write a word backwards

Now that we understand a bit about *while* loops, lets consider how a *while* loop could be applied to write a word or phrase backwards. Consider the following word



T	A	C	O	C	A	T
0	1	2	3	4	5	6

To get the last letter, the second to the last letter, the third to the last letter, etc of the word we can use the following approach,

```
var word = "TACOCAT";

var wordL = word.length();
var lastLetter = word.substring(wordL – 1, wordL);//last letter
var secondToLast = word.substring(wordL – 2, wordL – 1);//second to last
var thirdToLast = word.substring(wordL – 3, wordL – 2);//third to last
```

Consider the word declared above. Write a loop that could be used to store the reversal of the word in variable wordR. Then print wordR to the consol.

Now that you have the reversed word stored in its own variable, we can determine whether the word is a palindrome. A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam or racecar.

Write code that could be used to determine whether *word* and *wordR* are palindromes. If they are print to the console “Palindrome!”, otherwise print “Not a palindrome!”.

☐ Write a *while* loop to print a number backwards

In a previous lesson you wrote code to print out a number backwards. The problem however was that the number could only be four digits. *While* loops can be used to fix this issue and expand the problem to any number of digits (within the range of an integer of course)

Consider the number 1234567892 write a function that accepts a number as a parameter then reverses and returns the number.
Write another function that accepts a number as a parameter. In the body of this function, call the function you wrote above, then check whether the original number and the reversed number are the same – that is, whether the original number is a palindrome. If the number is a palindrome, your function should return true.

□ Write a *while* loop to count the number of heads in a given number of coin flips

Consider a simulator that keeps track of how many coin flips it takes to achieve a specified streak of heads. For example, how many coin flips does it take to get 3 heads in a row, 5 heads in a row, or even 12 heads in a row?

Write a function that accepts a number as a parameter. The parameter represents the number of heads. In the body of the function, determine the number of flips it takes to achieve the number of heads specified in row, Your function should return the total number of flips.

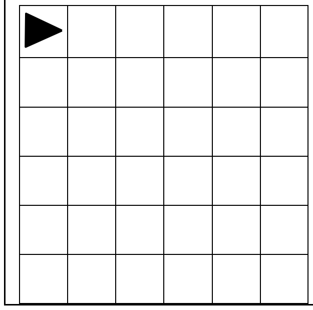
□ Write nested *while* loops to create shapes on a grid

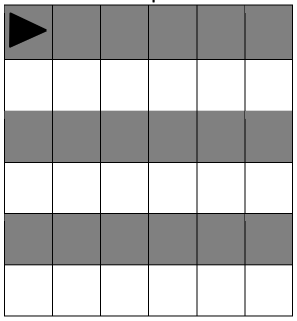
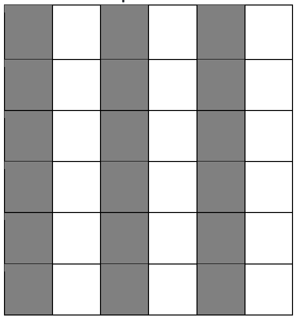
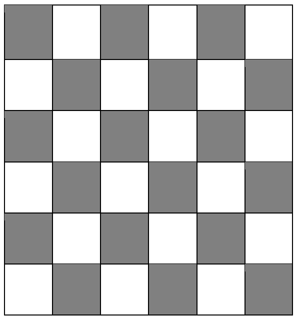
Nested loop is the term used when one loop is placed inside another as in the following example,

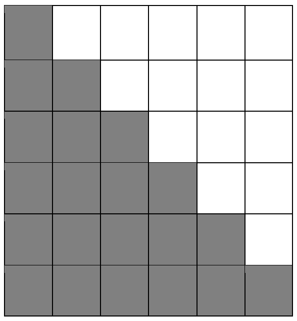
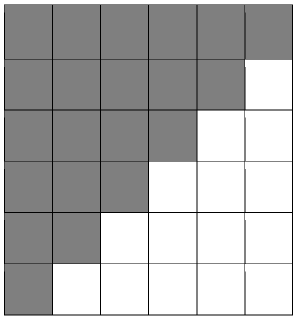
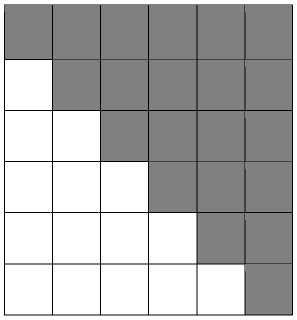
```
var j = 0, k = 0;
while(j < 5){
     console.log("outer loop");//executes 5 times
     while(k < 8){
          console.log("inner loop");//executes 5*8, or 40 times
          k++;
     }
    j++;
}
```

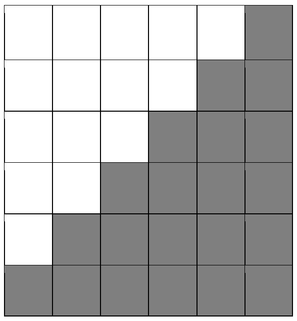
In the example above, the inner loop executes 8 times for each of the five iterations of the outer loop. Therefore, the code inside the inner loop will execute 40 times.

Consider the grid below,



Write code that could be used to create the following patterns,	
horizontalStripes 	
verticalStripes 	
checkerBoard 	

southWest 	
northWest 	
northEast 	

southEast 	

☐ 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.