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

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

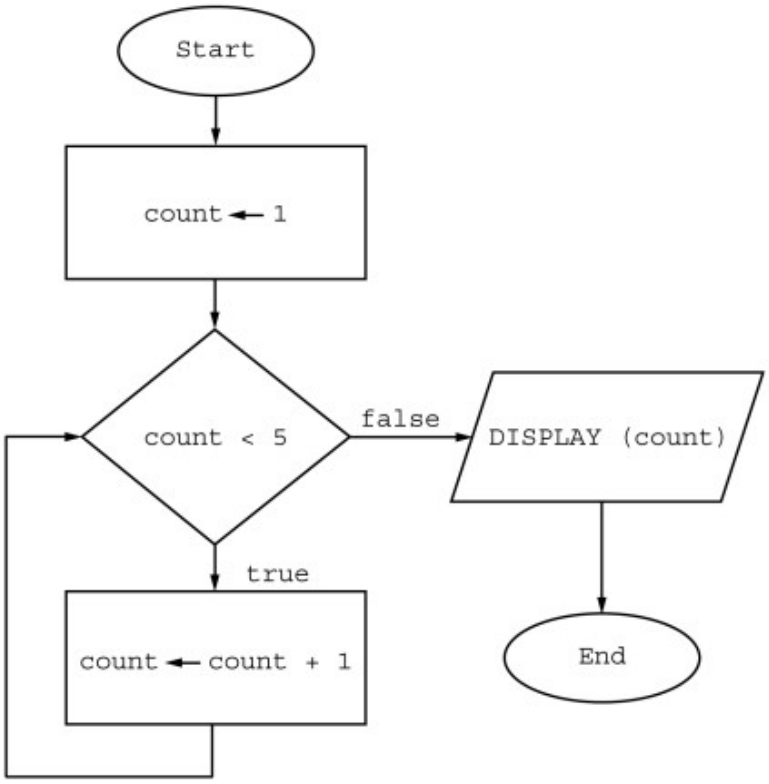
# While Loops

## Your Tasks (Mark these off as you go)

- ☐ Interpret loops in program flow charts
- ☐ Interpret *while loop* pseudocode
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Interpret nested *while loop* pseudocode
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Indentify an infinite *while loop*
- ☐ Write a *while loop*
- ☐ *Have Ms. Pluska check off the above tasks*
- ☐ Brainstorm a program
- ☐ Receive credit for the group portion of this lab

## ☐ Interpret loops in program flow charts

A loop is a data structure that allows us to repeat a block of code until a specified condition is met. The flow chart below illustrates the use of a loop to increment the variable *count*. Notice that in the example, *count* gets incremented until it reaches the value of 5, after which time the loop is exited and the result is displayed.




What is displayed as a result of executing the algorithm in the flowchart?

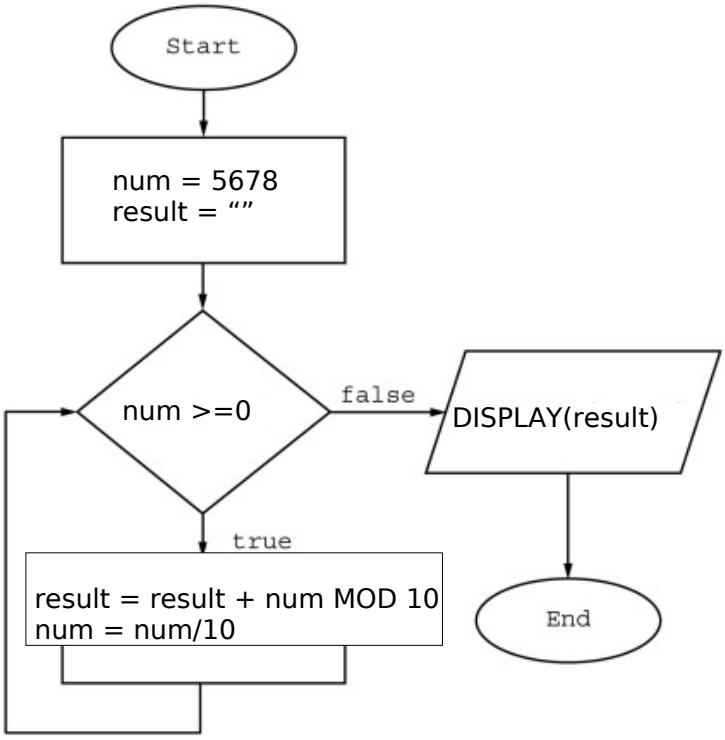
(A) 5

(B) 15

(C) 1 2 3 4

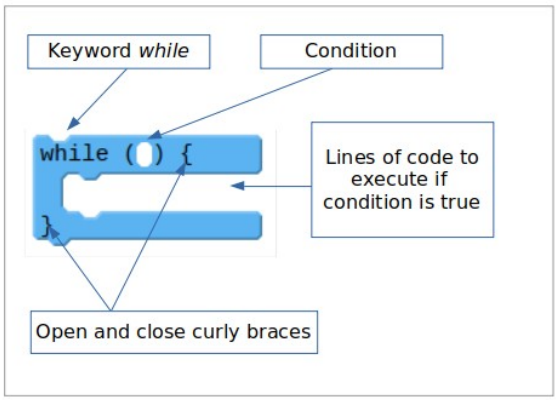
(D) 1 2 3 4 5

What is displayed as a result of executing the algorithm in the flowchart?




## ☐ Interpret while loop pseudocode

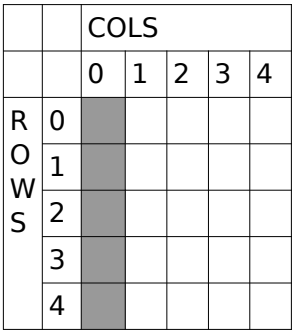
The *while* loop uses a *boolean* condition to repeated
expression, and if it is true it runs the block of code
checking the condition and running the block of cod
condition remains true. Once the boolean expressio
is a diagram showing the elements of a basic *while*




Before we get started writing *while loops*, lets practi
Although the examples below do use the while key
same way. That is, if a condition is true, keep loopir

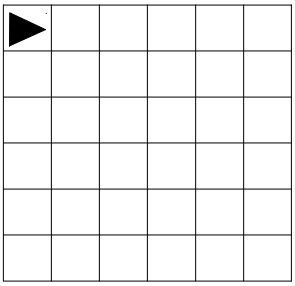
## EXAMPLE: Simple while loop

```
row = 0;

WHILE(row <= 4){
    MOVE_TO[row][col]
    FILL(grey)
    row = row + 1
}
```

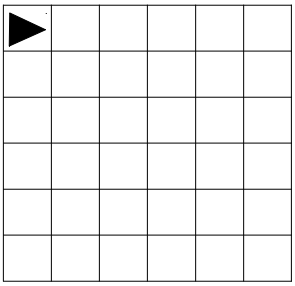



## YOU TRY IT - Simple while loop

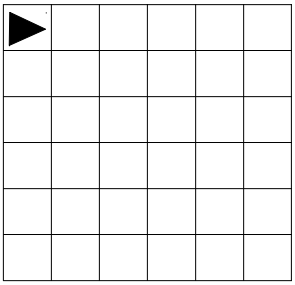
```
row = 0;
col = 0;

WHILE(row <= 5){
    FILL(grey)
    row = row + 1
    col = col + 1
    MOVE_TO[row][col]
}
```



```
row = 0;
col = 0;

WHILE(row <= 5){
    if((row MOD 2)EQUALS(0)){
       FILL(grey)

    }
    row = row + 1
    col = col + 1
    MOVE_TO[row][col]
}
```



```
row = 0;
col = 0;

WHILE(col <= 5){
    if((row MOD 2)EQUALS(0)){
       FILL(grey)
    }
    col = col + 1
    MOVE_TO[row][col]
    if(col == 5){
        row = row + 1
        col = 0
    }
}
```



In the procedure Mystery below, the parameter number is a positive integer. The procedure continues *while number* is less than or equal 0.

```
PROCEDURE Mystery (number)
{
   REPEAT UNTIL (number ≤ 0)
   {
      number ← number - 2
   }
   IF (number = 0)
   {
      RETURN (true)
   }
   ELSE
   {
      RETURN (false)
   }
}
```

Indicate the out put for each of the following calls.

(a) Mystery(2)

(b) Mystery(3)

(c) Mystery(4)

A program is created to perform arithmetic operations on positive and negative integers. The program contains the following incorrect procedure, which is inteded to return the product of the integers x and y. The loop "REPEAT UNTIL (count = y)" continues while count is not equal to y.

```
PROCEDURE Multiply (x, y)
{
   count ← 0
   result ← 0
   REPEAT UNTIL (count = y)
   {
      result ← result + x
      count ← count + 1
   }
   RETURN (result)
}
```

A programmer suspects that an error in the program is caused by this procedure. Under which of the following conditions will the procedure NOT return the correct product?

Select two answers.

(A) When the values of x and y are both positive.

(B) When the value of x is positive and the value of y is negative.

(C) When the value of x is negative and the value of y is positive.

(D) When the values of x and y are both negative.

In a certain science experiment, 75 percent of trials are expected to be successful and 25 percent of trials are expected to be unsuccessful. The program below is inteded to simulate the result of repeated trials of the experiment. The loop "REPEAT 1000 TIMES" continues while TIMES is not equal to 1000.

```
successful ← 0
unsuccessful ← 0
REPEAT 1000 TIMES
{
   IF (<MISSING CODE>)
   {
      successful ← successful + 1
   }
   ELSE
   {
      unsuccessful ← unsuccessful + 1
   }
}
DISPLAY (successful)
DISPLAY ("trials were successful,")
DISPLAY (unsuccessful)
DISPLAY ("trials were unsuccessful.")
```

Which of the following can be used to replace <MISSING CODE> so that the simulation works as intended?

(A) RANDOM (1, 100) = 25

(B) RANDOM (1, 100) ≤ 25

(C) RANDOM (1, 100) = 75

(D) RANDOM (1, 100) ≤ 75

## ☐ Have Ms. Pluska check off the above tasks


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

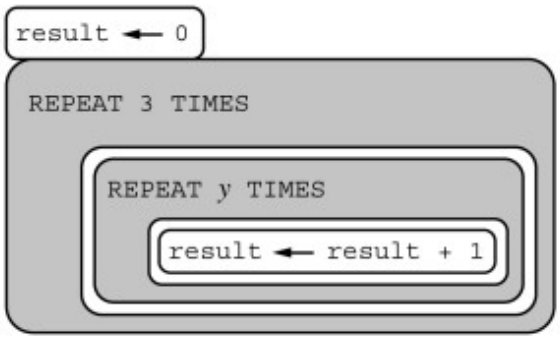
Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

## ☐ Interpret nested *while loop* pseudocode

While loops can also appear inside a while loop! In the example to the right, the the outer loop repeats 5 times and the inner loop repeats 2 times for a total of 5 x 2 = 10 times! This is because the outer loop cannot continue until the innerLoop has completed executing. So, when the outerLoop is 0, the innerLoop executes 2 times, when the outerLoop is 1, the innerLoop executes 2 times, so on and so forth, so the total count equals 2 x 5, or 10 at the end.

```
var outerLoop = 0;
var innerLoop = 0;
var count = 0;
while ( (outerLoop < 5) ) {
  while ( innerLoop < 2 ) {
    count++;
    innerLoop++;
  }
  innerLoop = 0;
  outerLoop++;
}
console.log(count);
```

In the program below, $y$ is a positive integer (e.g., 1, 2, 3, ...).




What is the value of `result` after running the program?

(A) $y + 3$

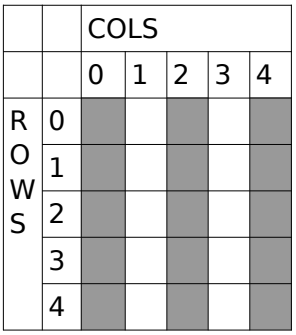
(B) $3y$

(C) $y^3$

(D) $3^y$

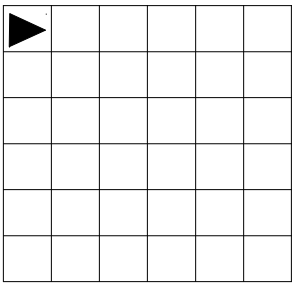
## EXAMPLE: Nested while loops

```
row = 0;
col = 0;

WHILE(row <= 4){
    WHILE(col <=4){
        MOVE_TO[row][col]
        if((col MOD 2)EQUALS(0)){
            FILL(grey)
        }
        col = col + 1;
    }
col = 0
row = row + 1
}
```

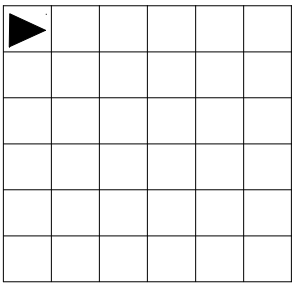



## YOU TRY IT – Nested while loops

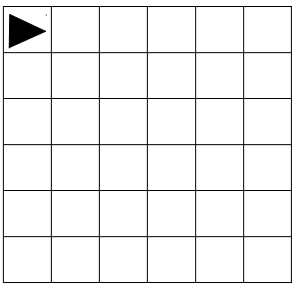
```
row = 0;
col = 0;

WHILE(row <= 5){
    WHILE(col <=5){
        MOVE_TO[row][col]
        if((col MOD 2)EQUALS(0)){
            FILL(grey)
        }
        col = col + 1
    }
col = 0
row = row + 2
}
```



```
row = 0;
col = 0;

WHILE(row <= 5){
    WHILE(col <=row){
        MOVE_TO[row][col]
        FILL(grey)
        col = col + 1
    }
col = 0
row = row + 1
}
```



```
row = 5;
col = 0;

WHILE(row >= 0){
    WHILE(col <= row){
        MOVE_TO[row][col]
        FILL(grey)
        col = col + 1
    }
col = 0
row = row - 1
}
```



## ☐ Have Ms. Pluska check off the above tasks


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

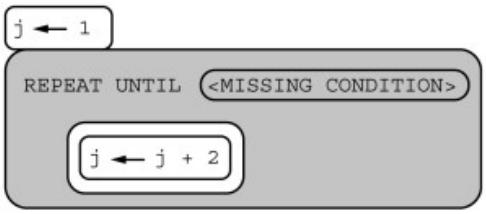
Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

## ☐ Identify an infinite *while loop*

*while* loops run until their condition becomes false, which raises an interesting question. What happens if the condition never becomes false? In these cases the program enters what is called an infinite loop over the commands in the *while loop*, and it never reaches the rest of your program. In the code below, what if the line indicated in the box was instead changed to `number ← number + 2`? If this were true, the procedure would never end, in fact your program would crash! For this reason, we normally avoid infinite loops in our programs.

```
PROCEDURE Mystery (number)
{
    REPEAT UNTIL (number ≤ 0)
    {
        number ← number - 2
    }
    IF (number = 0)
    {
        RETURN (true)
    }
    ELSE
    {
        RETURN (false)
    }
}
```

Consider the following code segment.




Which of the following replacements for `<MISSING CONDITION>` will result in an infinite loop?

(A) `j = 6`

(B) `j ≥ 6`

(C) `j = 7`

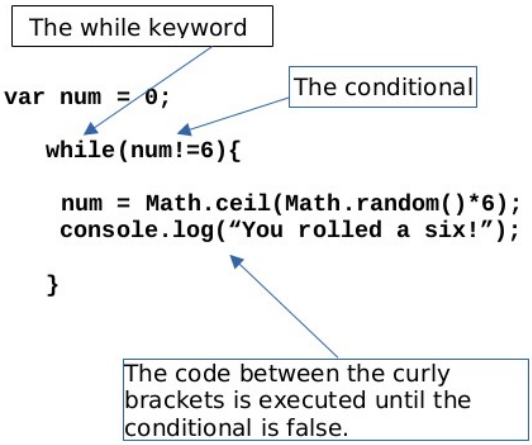
(D) `j > 7`

The given code accidentally loops infinitely, so something must be wrong with the condition. Can you figure out how to fix it?

```
var die1 = -1;
while ( die1 != 2 || die1 != 3 ) {
  die1 = randomNumber(1, 6);
  write( "Rolled a " + die1 );
}
write("Done.");
```

## ☐ Write a *while loop*

The syntax for writing a while loop in javascript is shown to the right. Just as we saw with the pseudocode examples, the *while loop* uses a *boolean* condition to repeatedly run a block of code. It checks the expression, and if it is true it runs the block of code contained within it. This process of checking the condition and running the block of code is repeated as long as the *boolean* condition remains true. Once the boolean expression becomes false it will stop.




Write a function called *coinFlip* which simulates the flipping of a coin. *coinFlip* should accept a parameter which represents the number of flips, then return the number of heads that result.

| |
|---|
| Write a function called *reverseNum* that accepts a number as a parameter, then returns the reversed number. For example, the following call would return 98765<br><br>`reverseNum(56789);` |
| |

## ☐ 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 _____

## ☐ Brainstorm a program

Challenge 1

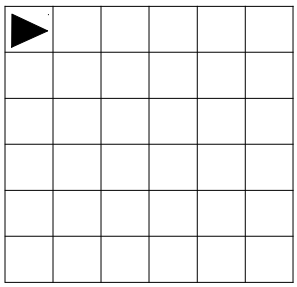
For your first challenge you will brainstorm a coin flipping simulator. The simulator will keep track of how many coin flips it takes to acheive a specified streak of heads. For example, how many coin flips does it take to get 3 heads in a row, 5 heads in a row, or even 12 heads in a row?

Challenge 2

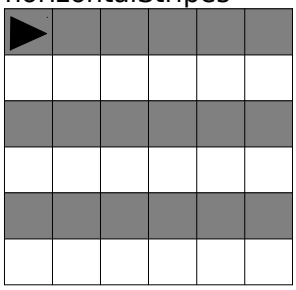
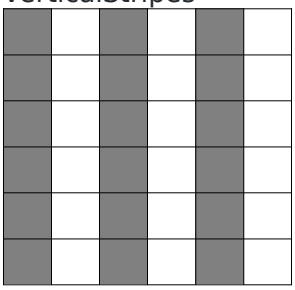
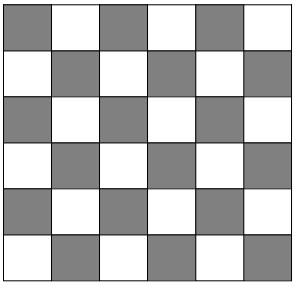
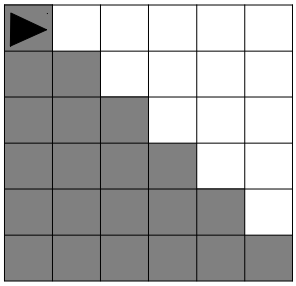
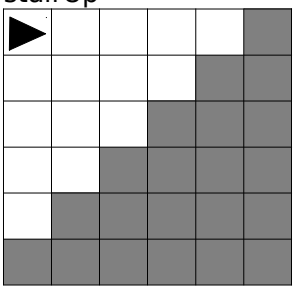
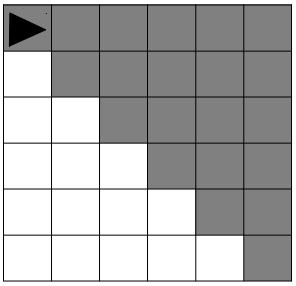
Brainstorm a program that prompts the user for a number base (2 thru 9) and a corresponding number, then converts the number to decimal

Challenge 3

Consider the grid below,



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

| | | |
|---|---|---|
| horizontalStripes  | verticalStripes  | checkerBoard  |
| stairDown  | stairUp  | stairDown2  |

Write your pseudocode on a separate sheet of paper. Your code should be legible and "make sense" to receive credit.

## ☐ Receive Credit for the group portion of this lab


- 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

_____