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

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

If Statements

Your Tasks (Mark these off as you go)

- ☐ Create an if statement project folder
- ☐ Interpret if statement pseudocode
- ☐ Have Ms. Pluska check off Interpret if statement pseudocode
- ☐ Write an if statement
- ☐ Write a complex if statement
- ☐ Have Ms. Pluska check off istatements and complex if statements
- ☐ Complete challenges 1 thru 2
- ☐ Receive credit for the group portion of this lab
- ☐ Receive credit for the individual portion of this lab

☐ Create an if statement project folder

This lab will follow the same workflow as the last lab. You will begin by making a new project directory within which you will create an index.html file and an app.js file. To view the results of your JavaScipt code, you will be using console. If you forgot how to do this, refer to the first lab "Introduction to JavaScript".

- ➔ First create a new folder on your computer called IfStatements.
- ➔ Add two new files to this folder,
  - o Index.html
  - o App.js

In your Index.html file, add the following code

Index.html

```
<html>
      <head>
           <script src = "App.js"></script>
     </head>
</html>
```

☐ Interpret if statement pseudocode

Each row in the table below presents a small program that uses if-statements and robot commands. Trace the code and plot the movements of the robot for the 3 scenarios shown to the right of the code. If the robot is directed to move onto a black square, it “crashes” and the program ends. If the robot doesn’t crash, then draw a triangle showing its ending location and direction.

There are a few patterns to the ways if-statements are typically used:

- ☐ Basic If-statements
- ☐ Sequential If-statements
- ☐ Nested If statements
- ☐ Combinations of all of the above

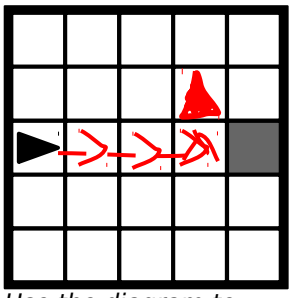
Each section below presents an example of one of these common patterns, followed by a few problems for you to try. For each type **study, and make sure you understand, the example** and why each of the 3 scenarios ends up in the state shown.

EXAMPLE: Basic If-statement

Code is executed sequentially from top to bottom. The code inside the if-block executes ONLY if the condition is true, otherwise the block is skipped and execution picks up on the first line after the if-block.

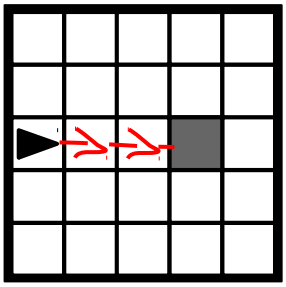
```
MOVE_FORWARD ()
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
  MOVE_FORWARD ()
}
ROTATE_LEFT ()
MOVE_FORWARD ()
```

Scenario 1:

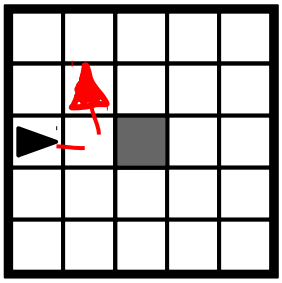


Use the diagram to trace each robot move.

Scenario 2:

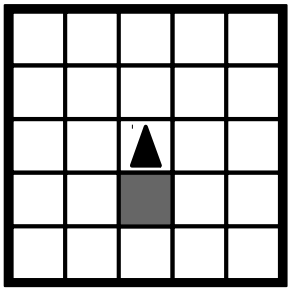


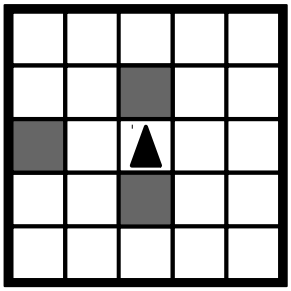
Scenario 3:

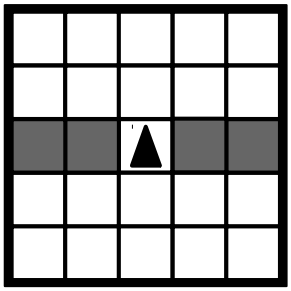


YOU TRY IT - Basic If-statement

```
ROTATE_LEFT ()
IF (CAN_MOVE (left))
{
  ROTATE_LEFT ()
}
MOVE_FORWARD ()
MOVE_FORWARD ()
```



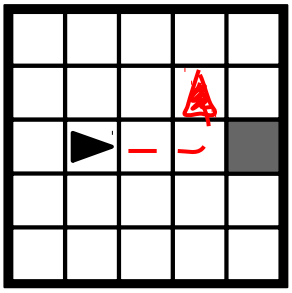


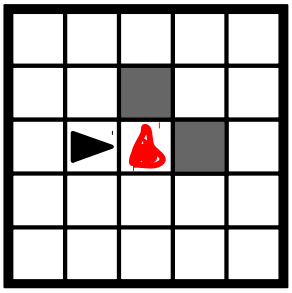


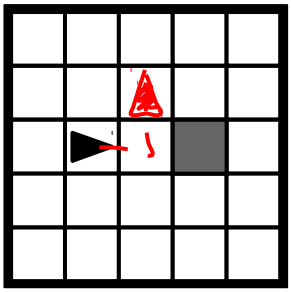
EXAMPLE: Sequential If-statements

Lines of code, including if statements, are evaluated separately, one at a time, in order from top to bottom. An if-block executes ONLY if the expression is true. Note that an earlier if-statement might change the state of the of world for an if-statement that comes later. This makes it hard to predict what will happen unless you trace the robot moves and take each line one at a time.

```
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
ROTATE_LEFT ()
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
```

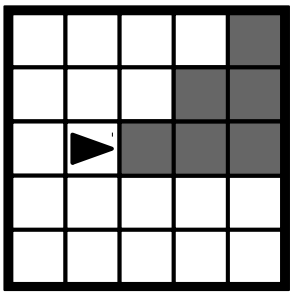
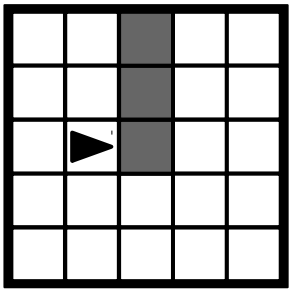
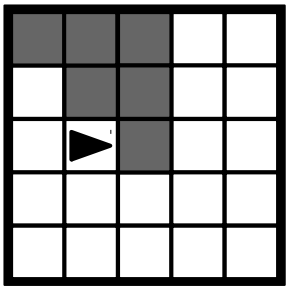




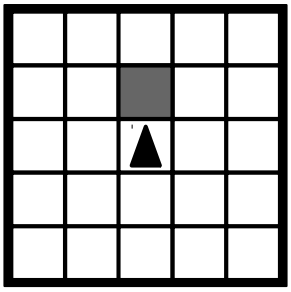
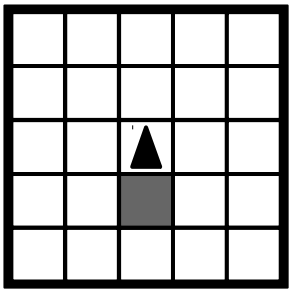
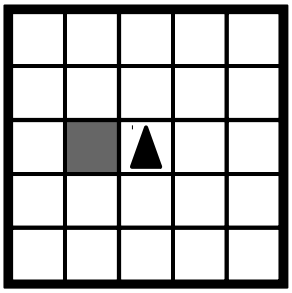


YOU TRY IT - Sequential If-statements

```
ROTATE_LEFT ()
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
ROTATE_RIGHT ()
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
ROTATE_LEFT ()
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
}
```

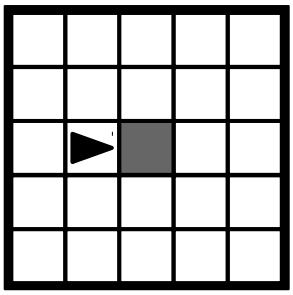
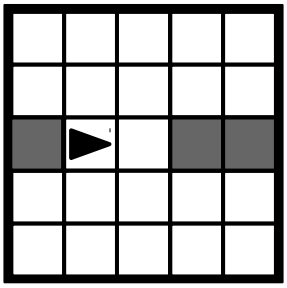
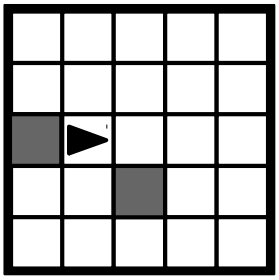




```
IF (CAN_MOVE ( left ))
{
  ROTATE_LEFT ()
  MOVE_FORWARD ()
}
IF (CAN_MOVE ( left ))
{
  ROTATE_LEFT ()
  MOVE_FORWARD ()
}
IF (CAN_MOVE ( left ))
{
  ROTATE_LEFT ()
  MOVE_FORWARD ()
}
```

YOU TRY IT - Sequential If-statements

```
IF (CAN_MOVE (forward))
{
  MOVE_FORWARD ()
  IF (CAN_MOVE (left))
  {
    ROTATE_LEFT ()
    IF (CAN_MOVE (right))
    {
      ROTATE_RIGHT()
    }
  }
}
MOVE_FORWARD ()
```

☐ Have Ms. Pluska check off Interpret if statement pseudocode

Before you continue have Ms. Pluska check off Interpret if statement pseudocode

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

☐ Write an if statement

In javascript, if statements take the following form.

```
if(true){
     console.log(“This message will print!”);
}
//Prints “This message will print!”;
```

Notice in the example above, we have an if statement. The if statement is composed of:

- The if keyword followed by a set of parentheses () which is followed by a code block, or block statement, indicated by a set of curly braces {}.
- Inside the parentheses (), a condition is provided that evaluates to true or false.
- If the condition evaluates to true, the code inside the curly braces {} runs, or executes.
- If the condition evaluates to false, the block won't execute.

Let's make some if statements!

➔ Using the let keyword, declare a variable named sale. Assign the value true to it.

➔ Now create an if statement. Provide the if statement a condition of sale. Inside the code block of the if statement, console.log() the string 'Time to buy!'.

➔ Notice that the code inside the if statement ran, since 'Time to buy!' was logged to the console. Below the sale variable declaration, but before the if statement, reassign sale to false. Run your code and observe what happens, we'll be changing this behavior in the next exercise.

➔ Consider the block of code below,

- Add an if-statement to the code to check the `age` to see if the person is old enough to drive. (In most states you need to be 16 or older).
- Display a message if the person is old enough drive.

```
console.log(“Driver Verification”);
var age = prompt(“Please enter your age”);
console.log(“It looks like you are old enough to drive!”);
console.log(“Thanks for verifying”);
```

☐ Write a complex if statement

If statements can include more than one boolean statement in the parentheses. For example, what if we wanted to evaluate the user name and password of a potential user? We could write the following,

```
if(username == un && password == pw){
     console.log(“You’re in!”);
}
```

In fact any of the boolean expressions you experimented with in the previous lesson can be placed be placed in between the parentheses following the if statement. For example,

```
var x = 79;
var y = 46;
var z = -3;
var w = 13.89;
var y = 40.0;

if(y/2 > w && w != x){
     console.log(“True”);
}
```

➔ Consider a number guessing program like the one you wrote previously.

➔ Prompt the user to pick a number between 0 and 100 (100 inclusive)

➔ If the number is greater than 0 and equal to or less than 100, the user can guess again. Otherwise the user can no longer guess.

☐ Have Ms. Pluska check off if statements and complex if statements

Before you continue have Ms. Pluska check off if statements and complex if statements

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

☐ Complete Challenges 1 thru 2

Challenge 1

Write a program that sorts numbers

➔ Begin your program by declaring three variables: low, middle, high;
➔ Next, declare a function called numberSort that accepts three parameters, n1, n2, and n3. The three paremeters represent numbers in no particular order.
➔ Write the numberSort function. The number sort function should identify the number with the lowest value and assign it to the variable low, identify the number with the highest value and assign it to the variable high, assign the number that is not the highest or the lowest to the the variable middle. The function should return the ordered numbers as list separated by commas.
  • For example the call below should return 3, 5, 10

```
numberSort(5, 3, 10);
```

➔ To test your program, prompt the user for three numbers. Call your function using the numbers provided by the user.
➔ Alert the user of the ordered list of numbers.

Challenge 2

Write a program that swaps the values of two variables

➔ Begin by declaring two variables called value1 and value2. You can initialize these variables to whatever you want for example value1 = “dog”, value2 = “cat”.
➔ Write a function that accepts two values and swaps the values.
  • For example the call below would assign value1 to cat and value2 to dog,

```
        swapValues(value1, value2);
```

☐ Receive Credit for the group portion of this lab

Make sure to indicate the names of all group members, then submit this lab to the needs to be graded folder to receive credit for the group portion of this lab.

☐ Receive Credit for the individual portion of this lab

Implement challenges 1 thru 2 on your computer. Show Ms. Pluska the completed challenges to receive credit for the individual portion of this lab.