

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

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

If-Else Statements

Your Tasks (Mark these off as you go)

- ☐ Create an if-else statement project folder
- ☐ Interpret if-else statement pseudocode
- ☐ Have Ms. Pluska check off Interpret if-else statement pseudocode
- ☐ Write an if-else statement
- ☐ Have Ms. Pluska check off writing if-else 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 else 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 JavaScript 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 IfElseStatements.
- ➔ Add two new files to this folder,
 - Index.html
 - App.js

In your Index.html file, add the following code

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

☐ Interpret if else statement pseudocode

Each row in the table below presents a small program that uses if-else 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. We explored several of these in previous lesson.

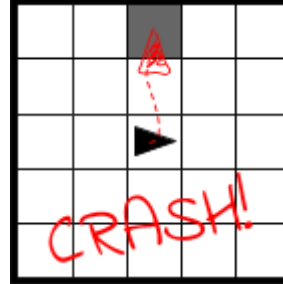
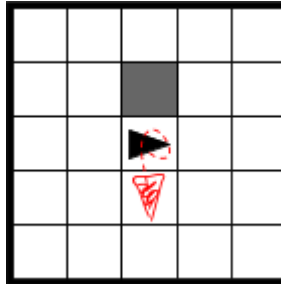
- ☐ Basic If-statements
- ☐ Sequential If-statements
- ☐ Basic if-else statements
- ☐ Nested If and if-else statements
- ☐ Combinations of all of the above
- ☐ Combinations of all of the above

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.

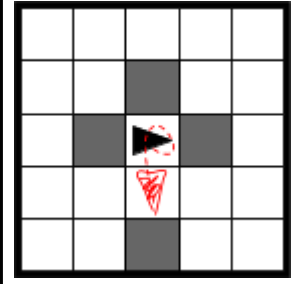
EXAMPLE: If-else Statement

The code in the if-block executes *ONLY* if the expression is true, otherwise the code in the else block will run. But one or the other *must* execute. An else statement can be attached to a single if-statement.

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

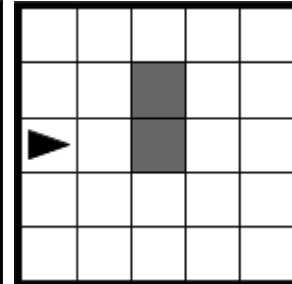
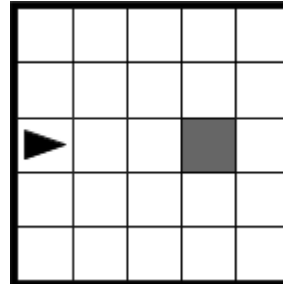
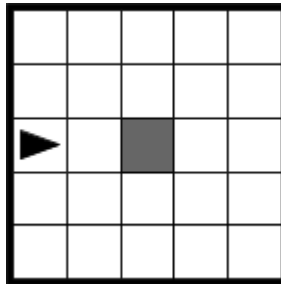
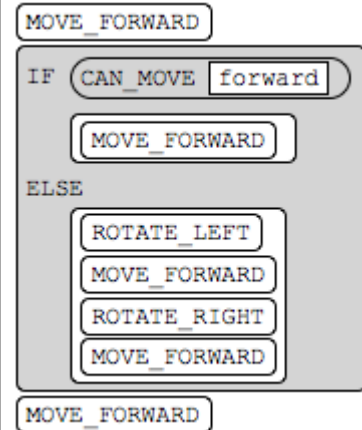


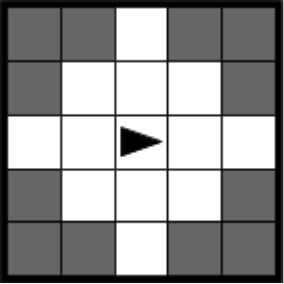
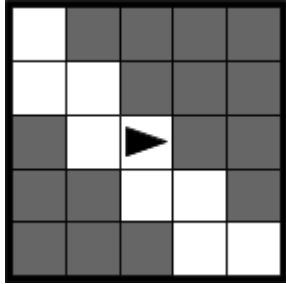
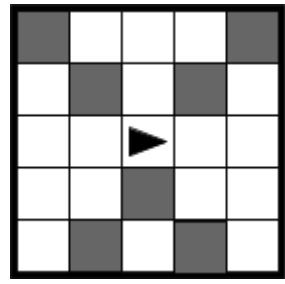
NOTE: Easy to miss
MOVE_FORWARD on
the last line

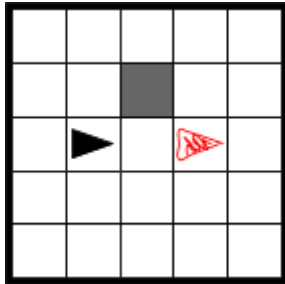
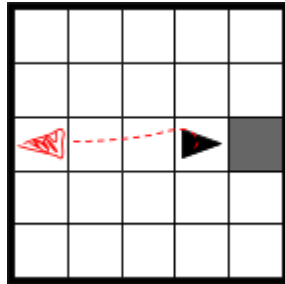
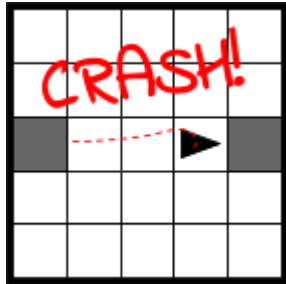


YOU TRY IT - Simple If-Else

Here is a block-based version of a very similar program.



<pre> IF (CAN_MOVE (left)) { ROTATE_LEFT () MOVE_FORWARD () } ELSE { ROTATE_RIGHT() MOVE_FORWARD () } IF (CAN_MOVE (right)) { ROTATE_RIGHT () } ELSE { ROTATE_LEFT () } MOVE_FORWARD () </pre>			
--	---	--	---

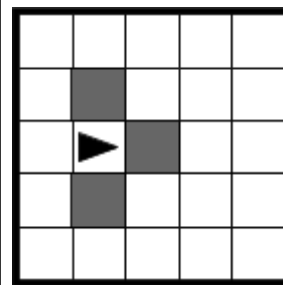
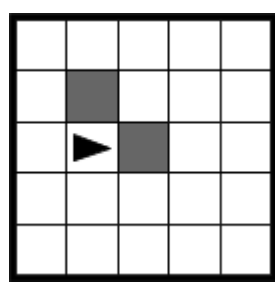
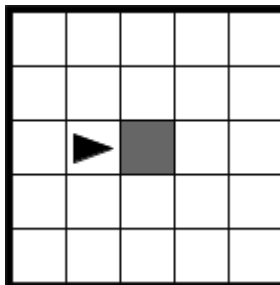
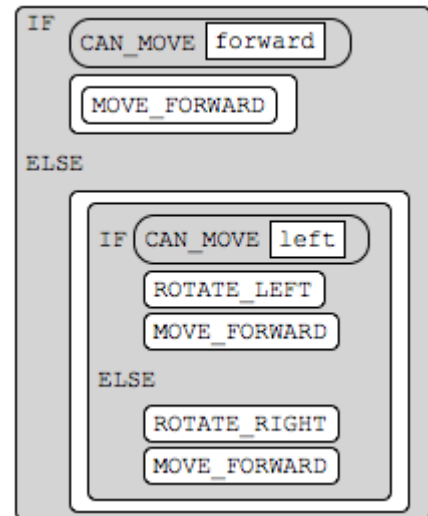
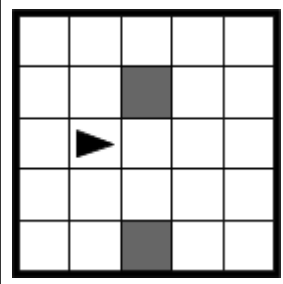
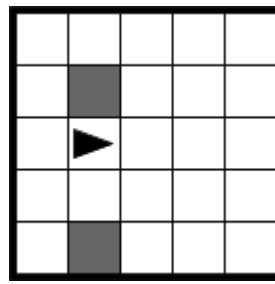
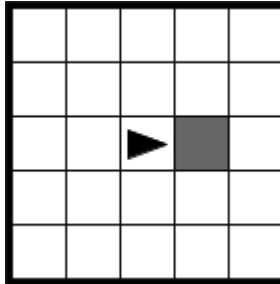
EXAMPLE: Nested Statements			
<p><i>You can put if- and if-else statements inside other if-statements. All previous rules apply, but tracing the code can be tricky.</i></p> <pre> IF (CAN_MOVE (forward)) { MOVE_FORWARD () } ELSE { IF(CAN_MOVE(backward)) { ROTATE_LEFT () ROTATE_LEFT () MOVE_FORWARD () } MOVE_FORWARD () } MOVE_FORWARD () </pre>			

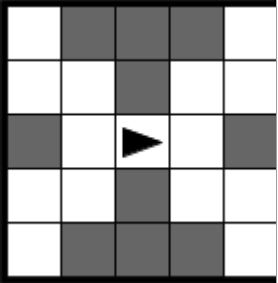
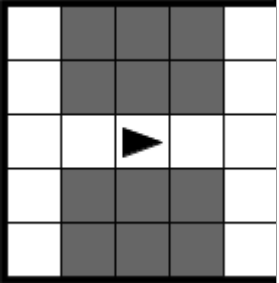
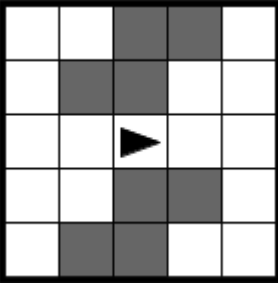
YOU TRY IT - Nested If-statements

```

IF (CAN_MOVE (forward))
{
  IF (CAN_MOVE (left))
  {
    ROTATE_LEFT ()
  }
  ELSE{
    ROTATE_RIGHT ()
  }
  MOVE_FORWARD ()
}
ELSE
{
  ROTATE_LEFT ()
  ROTATE_LEFT ()
}
MOVE_FORWARD ()

```



<pre> IF (CAN_MOVE (forward)) { MOVE_FORWARD () IF(CAN_MOVE (left)) { IF(CAN_MOVE (right)) { ROTATE_RIGHT () } ELSE { ROTATE_LEFT () } } ELSE { ROTATE_RIGHT () } MOVE_FORWARD () } ELSE { ROTATE_LEFT () ROTATE_LEFT () } MOVE_FORWARD () </pre>			
---	---	--	---

□ Have Ms. Pluska check off Interpret if-else statement pseudocode



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

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

□ Write an if-else statement

In javascript, if-else statements take the following form.

```

if(true){
  console.log("It is true");
}else{
  console.log("It is false:");
}

```

Notice in the example above, we have an if-else 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 code inside the curly braces {} following the else statement runs.

Let's make some if statements!

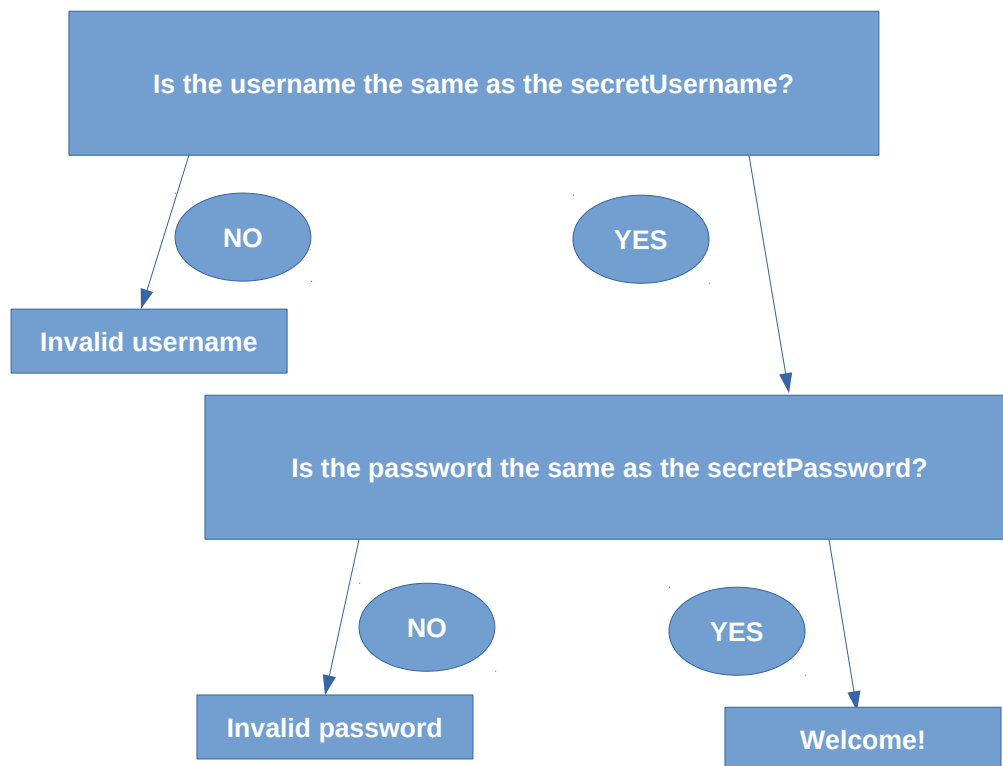
Part 1: Guess my number

In a previous lesson you wrote a “guess my number” program. The program was limited in that it could only return true or false. Modify the program using if-else statements.

- ➔ Declare and initialize a variable called secretNumber
- ➔ Prompt the user for a number and assign the number to a new variable called guess
- ➔ Write a function that accepts a parameter. The parameter represents the user's guess. Your function should compare the user's guess to the secretNumber.
 - If they are the same return “You got it!”
 - else return “Try again!”
- ➔ Prompt the user for at least 10 guesses. Each prompt should display “You got it!” or “Try again!”
- ➔ After the tenth guess alert the user of the secretNumber

Part 2: Create a login validator

You will create a simple login validator. A flow chart for how the validator should work is shown below. Notice, this task will require a nested if-else statement



- ➔ Declare and initialize a variable called secretUsername
- ➔ Declare and initialize a variable called secretPassword
- ➔ Create a prompt that prompts the user for their username, assign the value to a new variable called username
- ➔ Create a prompt that prompts the user for their password, assign the value to a new variable called password
- ➔ Write an if-else statement that compares the username to the secretUsername. If the username and secretUsername match, prompt the user for their password, otherwise alert the user that their username is invalid.

- ➔ Write an if-else statement that compares the password to the secretPassword. If the password and secretPassword match, alert the user "Welcome!", otherwise alert the user that their password is invalid
- ➔ 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.

□ Have Ms. Pluska check off if-else statements



Before you continue have Ms. Pluska check off if-else statements

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

□ Complete Challenges 1 thru 2

Challenge 1

Expand your guess my number program

- ➔ Begin by declaring and initializing a variable called secretNumber.
- ➔ Prompt the user to guess a number between 0 and 100
- ➔ Write a function that accepts a parameter which represents the user's guess. Your function should evaluate whether or not the guess is within the specified range. If the guess is within the specified range, evaluate whether the guess is too high, too low, or exact. Your function should return one of the following messages depending on the outcome,
 - Too low
 - Too high
 - You guessed it!
- ➔ Prompt the user at least 10 times for your secret number

Challenge 2

For this challenge you will create a variation of the Wheel of Fortune game. Your game will begin by prompting the user to guess the letters of a mystery word provided a clue.

Consider the following criteria when designing your program,

- The mystery word must be at least three letters long but no longer than four (Keep it short!)
 - The user can guess one letter at a time. After each guess, the program must indicate to the user whether or not the letter is in the word.
 - The user can guess the word at anytime. After each guess, the program must indicate to the user whether or not the word is correct.
 - The user should be allowed at least five guesses total.
 - After five guesses you must indicate the word and indicate to the user whether or not they were successful.
-
- ➔ Begin by declaring and initializing a variable secretWord;
 - ➔ Create a prompt that provides a clue to the user and prompts them for a letter. Assign the input provided by the user to a new variable called userGuess.
 - ➔ Write a function that accepts one parameter which represents the user's guess.
 - ➔ In the body of the function, evaluate whether or not the guessed letter is in the secret word, or whether the user identified the secret word.

- ➔ Continue to prompt the user and call the function to evaluate their guess at least five times
- ➔ After the five try, alert the user of the secret word

Below is an example of what this game might look like,

Prompt: Animal with four legs. Type a letter or animal to begin.

User input: C

Prompt: There are no C's. You have four tries remaining. Please guess another letter or animal.

User input: A

Prompt: There is one A. You have three tries remaining. Please guess another letter or animal.

User input: BAT

Prompt: BAT is incorrect. You have two tries remaining. Please guess another letter or animal.

User input: R

Prompt: There is one R. You have one try remaining. Please guess another letter or animal.

User input: RAT

Alert: Success! Animal with four legs is a RAT!

❑ 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.