|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **If Statements** |  |

|  |
| --- |
| **Your Tasks (Mark these off as you go)** |
| * Create an if statement project folder * Interpret if statement pseudocode * Apply the AND and OR operators * Have Ms. Pluska check off Boolean expressions and AND/OR operators * Group Boolean expression * Write functions that return Boolean values * Have Ms. Pluska check off grouping Boolean expressions and writing Boolean functions * 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,
  + 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 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
* Basic If-else statements
* Nested If and if-else 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 ()  } |  |  |  |

* **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 |

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 “= =” |

* In your app.js file declare the following variables,

var x = 79;

var y = 46;

var z = -3;

var w = 13.89;

var y = 40.0;

var t = true;

var f = false;

* What does each of the following evaluate to?

console.log(true && false);

console.log(true && !false);

console.log(!t || f);

console.log(x != 3 || f );

console.log(x == y || f);

console.log(y/2 > w && w != x);

* **Have Ms. Pluska check off Boolean expressions and AND /OR operators**



Before you continue have Ms. Pluska check off Boolean expressions and AND/OR operators

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) );

* In your app.js file, determine what each of the following evaluate to

console.log((x > 102) && true);

console.log((z == 1) || false);

console.log((z == 40) && !false);

* **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

In the last lesson, you wrote a function that calculated how many monitors we needed for an office shaped like a grid. This time, we will need to consider if we have enough money saved to actually purchase the monitors.

* Declare a function monitorCount() that has two parameters. The first parameter is rows and the second parameter is columns.
* 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.
* 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 grouping Boolean expressions and writing fuctions that return Boolean values**



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 \_\_\_\_\_\_\_\_\_\_\_\_

* **Complete Challenges 1 thru 2**

Challenge 1

Write a guess my number appliction.

* Begin by declaring a new variable called secret number. Assign a number to this value. It can be any number between 0 and 100.
* Next, create a prompt, that prompts ther user for a guess.
* Next create a function that returns a Boolean value. The function should accept two parameters, the secret number and the user’s guess. Your function should return the result (userGuess == secretNumber). DO NOT user if statements for this!
* Your final program should prompt the user for at least 10 guesses. After each guess, you must alert the user whether their guess is right or wrong. After the 10th guess, you should alert the user of the actual number.

Challenge 2

Write a math facts program.

* Begin by prompting the user for answers to the math facts... For example 5 x 5 = , 5 x 4 = , etc.
* Create a function that returns a Boolean value. The function should alert the user whether or not they got the correct answer to each fact.
* You may create additional variables or functions as needed, but you CANNOT USE if statements.
* **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.