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

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

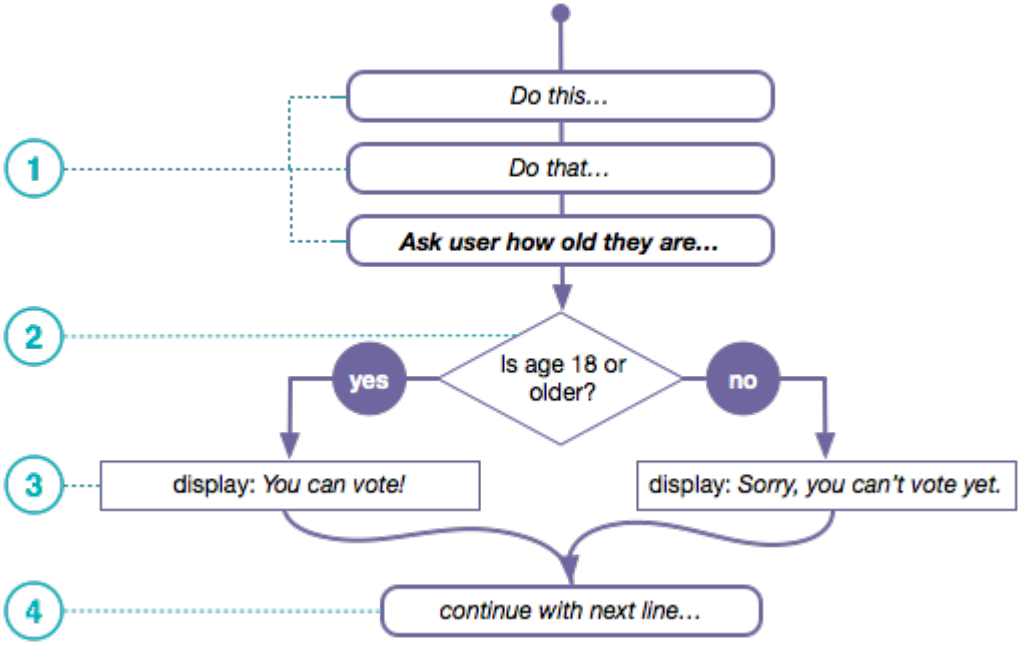
If Statements

Your Tasks (Mark these off as you go)

- ☐ Interpret program flow charts
- ☐ Interpret if statement pseudocode
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Write an if statement
- ☐ Write a complex if statement
- ☐ Have Ms. Pluska check off istatements and complex if statements
- ☐ Brainstorm a program
- ☐ Receive credit for the group portion of this lab

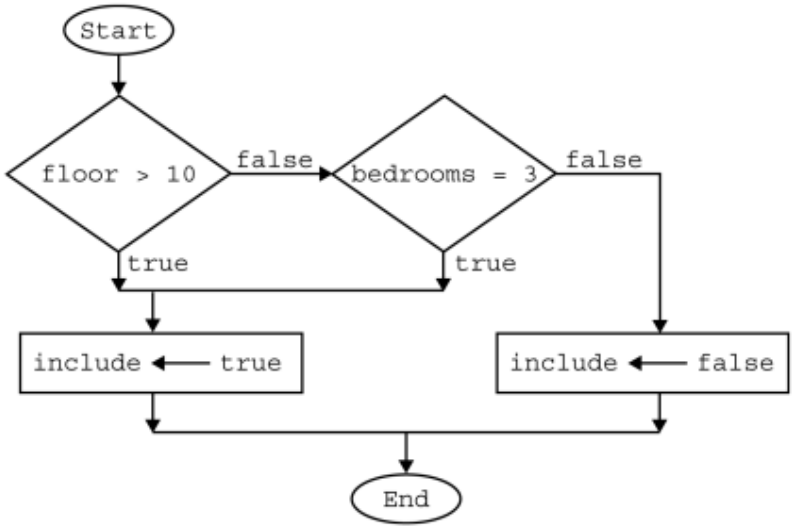
☐ Interpret program flow charts

Programs are said to have a "flow of execution". You start by executing some line of code and then the next and so on. A flow chart is a common visual that's used to represent the various paths of execution that your program might take. Many people use them to help plan. Below is an example.



1. The flow chart above depicts a program executing one line after another until it gets to a point where it needs to make a decision.
2. In order to determine which path to take you state some condition. It should be a Boolean expression - something that evaluates to *true* or *false*. Here we have a simple comparsion of two values: the person's age and the number 18.
3. The program does one thing if the condition is true, and something else if the condition is false.
4. The program can continue a single thread of execution after the condition as well.

Block	Explanation
Oval ⬭	The start or end of the algorithm
Diamond ◇	A conditional or decision step, where execution proceeds to the side labeled `true` if the condition is true and to the side labeled `false` otherwise
Rectangle ▭	One or more processing steps, such as a statement that assigns a value to a variable



Which of the following statements is equivalent to the algorithm in the flowchart?

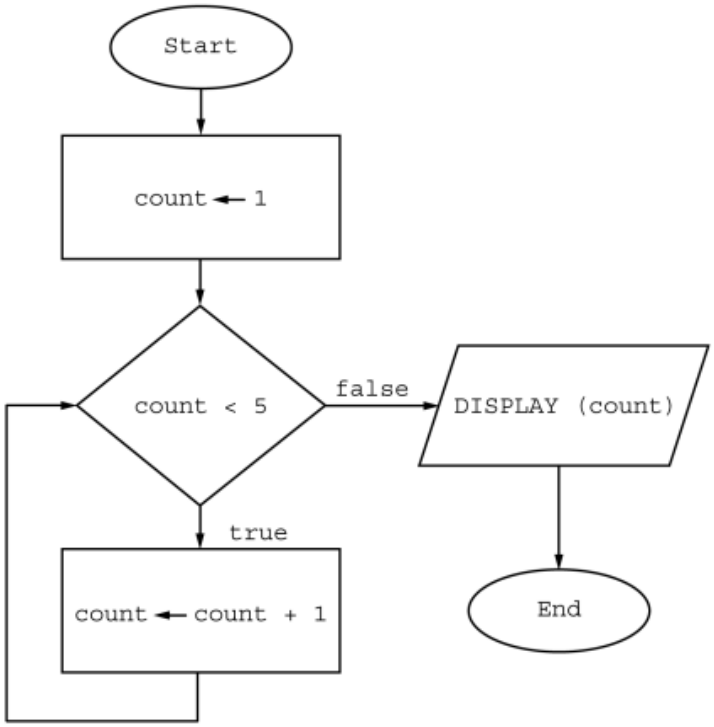
(A) `include ← (floor > 10) OR (bedrooms = 3)`

(B) `include ← (floor > 10) AND (bedrooms = 3)`

(C) `include ← (floor ≤ 10) OR (bedrooms = 3)`

(D) `include ← (floor ≤ 10) AND (bedrooms = 3)`

Block	Explanation
Oval	The start or end of the algorithm
Rectangle	One or more processing steps, such as a statement that assigns a value to a variable
Diamond	A conditional or decision step, where execution proceeds to the side labeled `true` if the condition is true and to the side labeled `false` otherwise
Parallelogram	Displays a message



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`

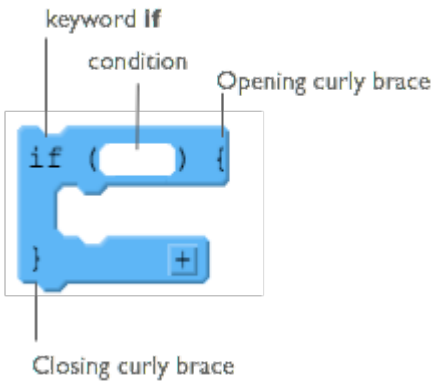
Central High School keeps a database of information about each student, including the numeric variables *numberOfAbsences* and *gradePointAverage*. The expression below is used to determine whether a student is eligible to receive an academic award.

```
(numberOfAbsences ≤ 5) AND (gradePointAverage > 3.5 )
```

Draw a flowchart to represent the statement above. If the conditions above are met, the variable *academicAward* is true, otherwise it is false.

☐ Interpret if statement pseudocode

if statements are the lines of code you need to change the flow while you're program is running. You can write code that makes a decision that determines which lines of code should be run next.



At the right is a diagram showing the elements of a basic *if* statement in JavaScript.

There are two basic parts to an if-statement.

1. A condition to be evaluated (A Boolean expression that evaluates to true or false)
2. Code that should run if the expression was true - enclosed in curly braces

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

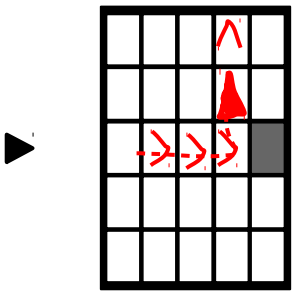
Each section below presents an example of one of these common patterns, followed by a few problems for you to try. For each type **study, and make sure you understand, the example** and why each of the 3 scenarios ends up in the state shown.

EXAMPLE: Basic If-statement

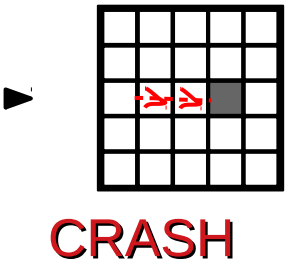
Code is executed sequentially from top to bottom. The code inside the if-block executes ONLY if the condition is true, otherwise the block is skipped and execution picks up on the first line after the if-block.

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

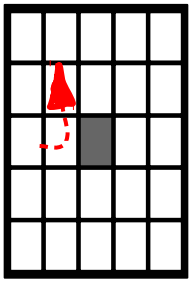
Scenario 1:



Scenario 2:

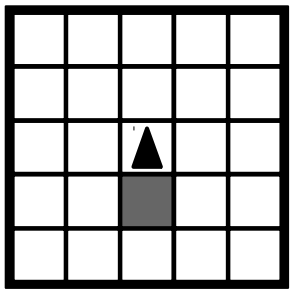


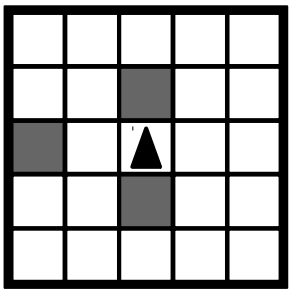
Scenario 3:

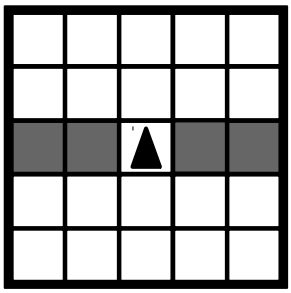


YOU TRY IT - Basic If-statement

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



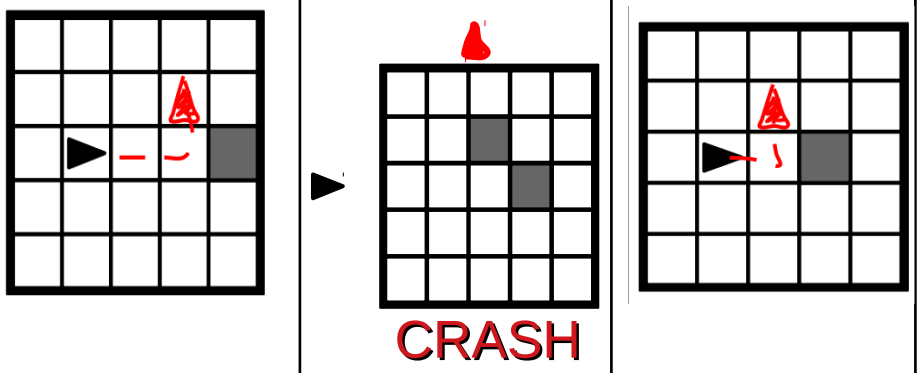




EXAMPLE: Sequential If-statements

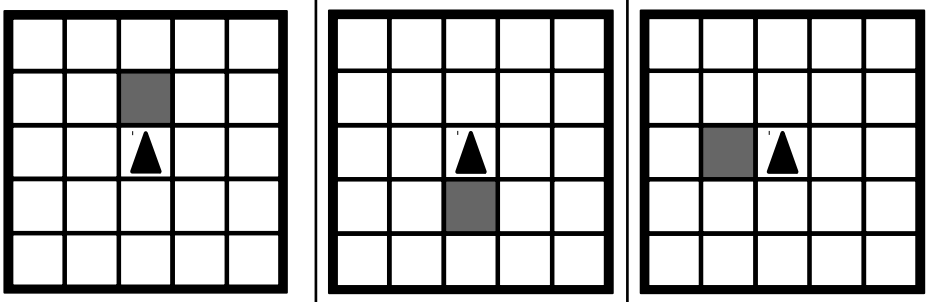
Lines of code, including if statements, are evaluated separately, one at a time, in order from top to bottom. An if-block executes ONLY if the expression is true. Note that an earlier if-statement might change the state of the of world for an if-statement that comes later. This makes it hard to predict 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 ()
}
```



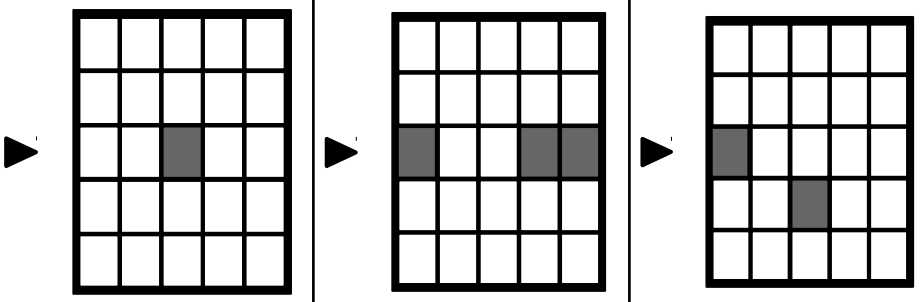
YOU TRY IT - Sequential If-statements

```
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 - Nested 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 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 _____

☐ 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.

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!'.

Consider the block of code below,

- Re-write the code and 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!");
```

☐ 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 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 the following students and their corresponding gpa's. Notice their rank is out of order! Write a program that puts the students in the correct order. The gpa and rank of each student can be accessed using the following syntax: Bart.gpa, Bart.rank

	gpa	**rank**
var Bart	3.5	1
var Bugs	3.8	3
var Kyle	3.1	2

☐ 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

Part 1

Consider a program that generates 2 buttons with the following values. Brainstorm an algorithm that could be used to swap the values that are associated with the buttons. That is, b1.id = 2 and b2.id = 9

9	2

Part 2

Now, consider a program that generates 7 buttons with the following values,

6	2	4	1	7	5	3

Notice that the values are out of order! Brainstorm a program that could be used to enable the user to put the buttons in order. For example when 6 and 1 are clicked the values that appear on these buttons are swapped.

Your final program should also include a timer which alerts the user the total time it took to sort the buttons!

☐ 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

_____