Name(s)_____ Period _____ Date _____

Activity Guide - Will it Crash?

Let's play a game: "Will it Crash?"

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

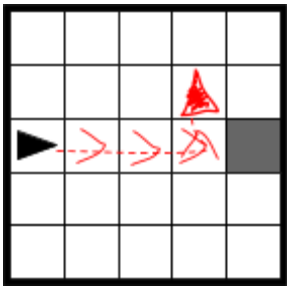
Each section below presents an example of one of these common patterns, followed by a few problems for you to try. For each type **study, and make sure you understand, the example** and why each of the 3 scenarios ends up in the state shown.

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.

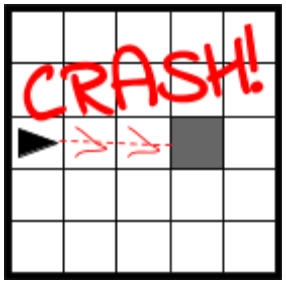
```
MOVE_FORWARD ()
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
   MOVE_FORWARD ()
}
ROTATE_LEFT ()
MOVE_FORWARD ()
```

Scenario 1:

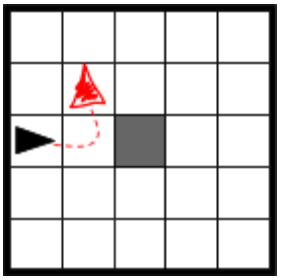


Use the diagram to trace each robot move.

Scenario 2:

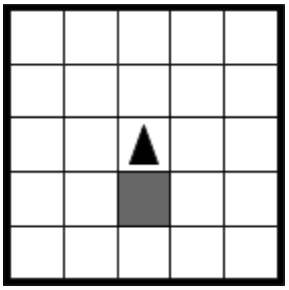


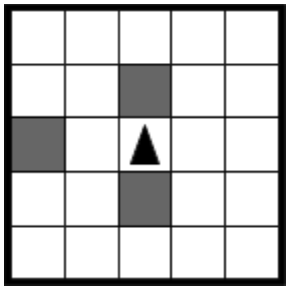
Scenario 3:

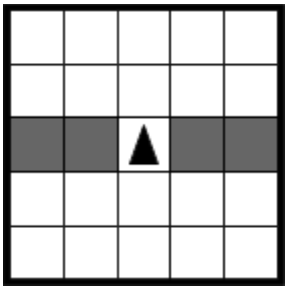


YOU TRY IT - Basic If-statement

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



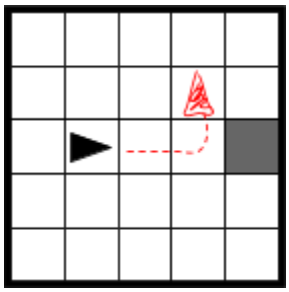


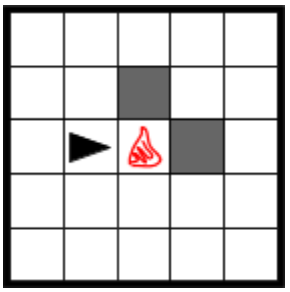


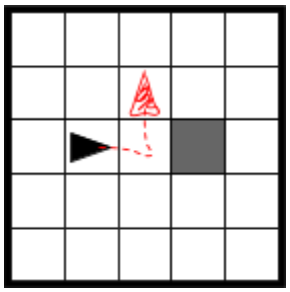
EXAMPLE: Sequential If-statements

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 ()
}
```

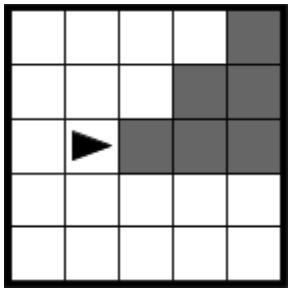


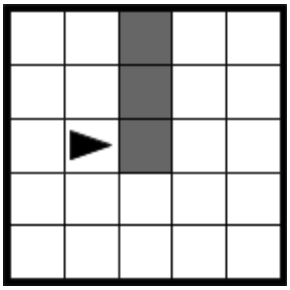


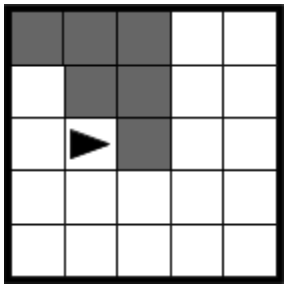


YOU TRY IT - Sequential If-statements

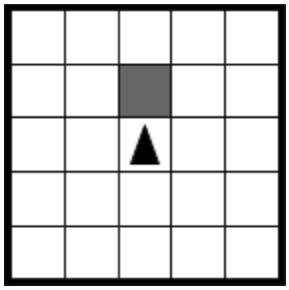
```
ROTATE_LEFT ()
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
}
ROTATE_RIGHT ()
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
}
ROTATE_LEFT ()
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
}
```

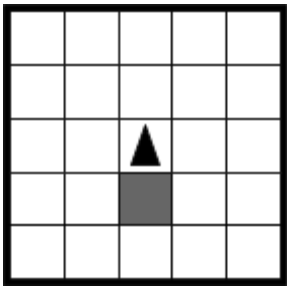


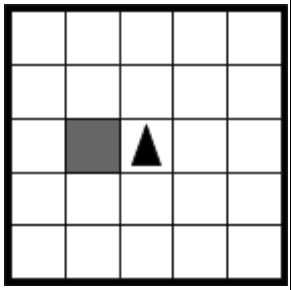




```
IF (CAN_MOVE ( left ))
{
   ROTATE_LEFT ()
   MOVE_FORWARD ()
}
IF (CAN_MOVE ( left ))
{
   ROTATE_LEFT ()
   MOVE_FORWARD ()
}
IF (CAN_MOVE ( left ))
{
   ROTATE_LEFT ()
   MOVE_FORWARD ()
}
```



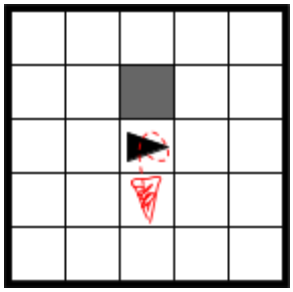


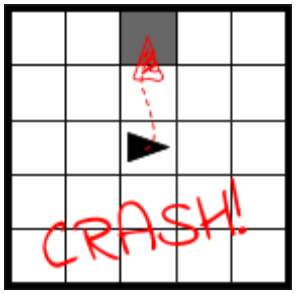


EXAMPLE: If-else Statement

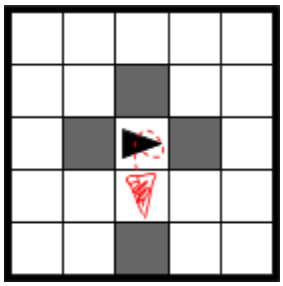
The code in the if-block executes ONLY if the expression is true, otherwise the code in the else block will run. But one or the other must execute. An else statement can be attached to a single if-statement.

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



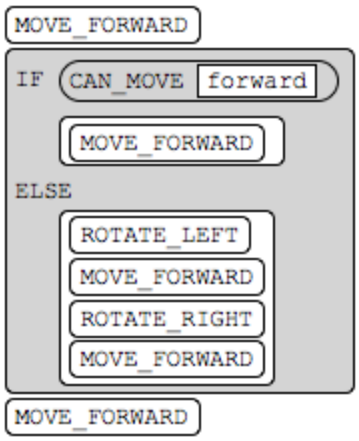


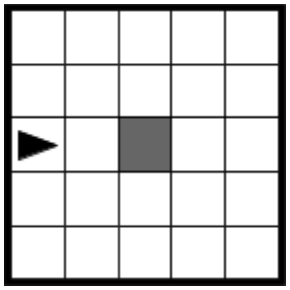
NOTE: Easy to miss MOVE_FORWARD on the last line

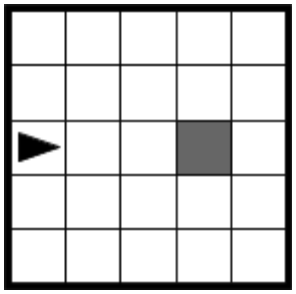


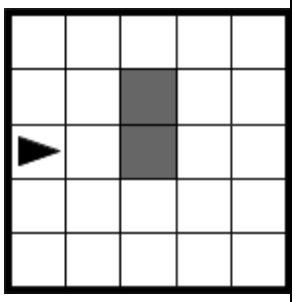
YOU TRY IT - Simple If-Else

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

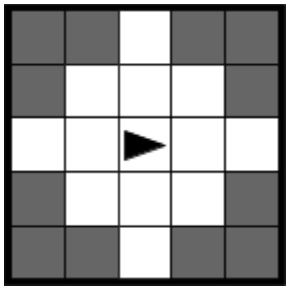


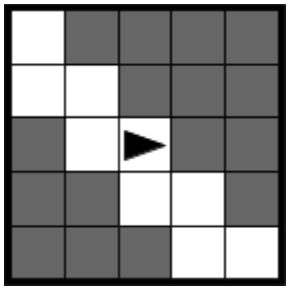


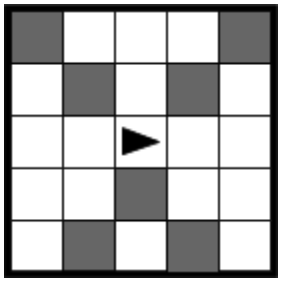




```
IF (CAN_MOVE (left))
{
   ROTATE_LEFT ()
   MOVE_FORWARD ()
}
ELSE
{
   ROTATE_RIGHT()
   MOVE_FORWARD ()
}
IF (CAN_MOVE (right))
{
   ROTATE_RIGHT ()
}
ELSE
{
   ROTATE_LEFT ()
}
MOVE_FORWARD ()
```



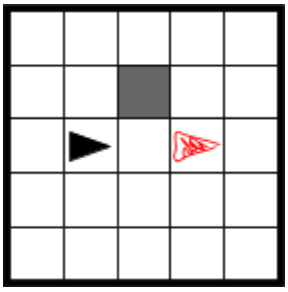


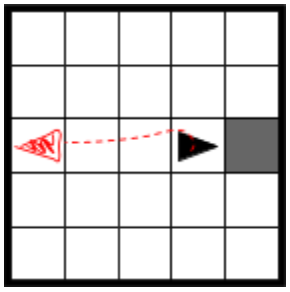


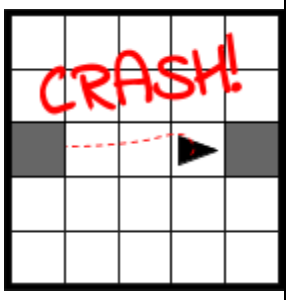
EXAMPLE: Nested Statements

You can put if- and if-else statements inside other if-statements. All previous rules apply, but tracing the code can be tricky.

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

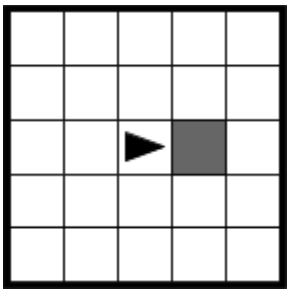


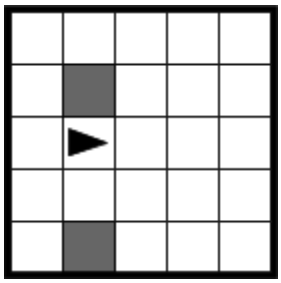


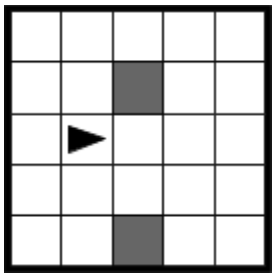


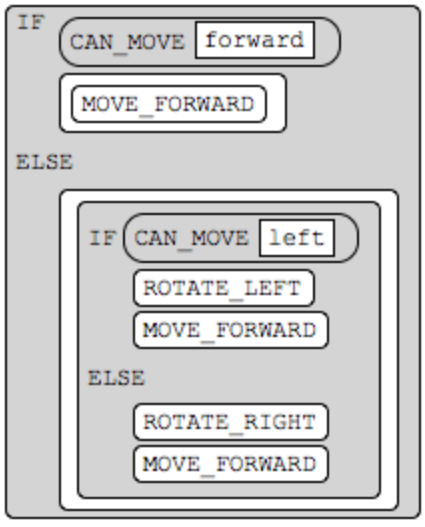
YOU TRY IT - Nested If-statements

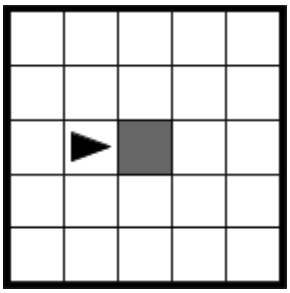
```
IF (CAN_MOVE (forward))
{
   IF (CAN_MOVE (left))
   {
      ROTATE_LEFT ()
   }
   ELSE{
      ROTATE_RIGHT ()
   }
   MOVE_FORWARD ()
}
ELSE
{
   ROTATE_LEFT ()
   ROTATE_LEFT ()
}
MOVE_FORWARD ()
```

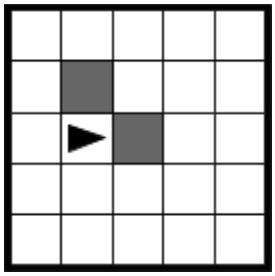


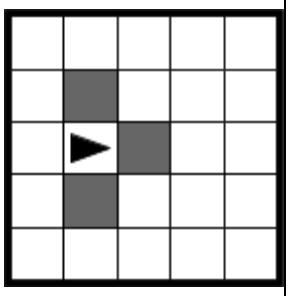




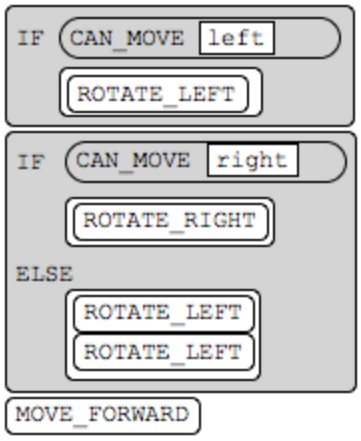


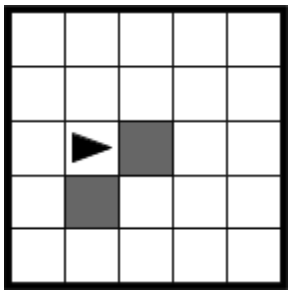


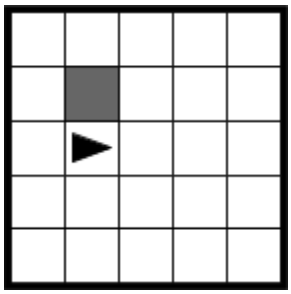


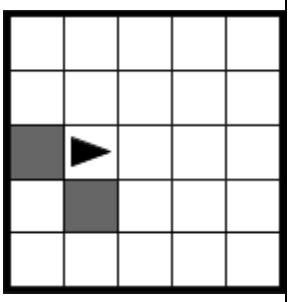


Challenge: putting it all together — if, if-else, sequential if, nested statements

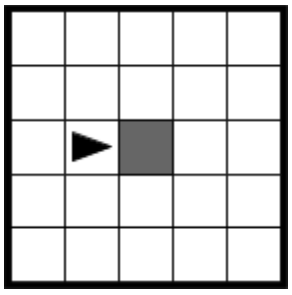


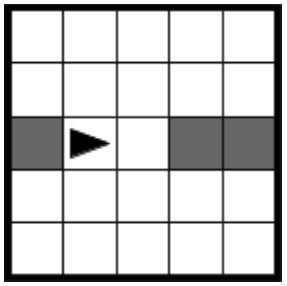


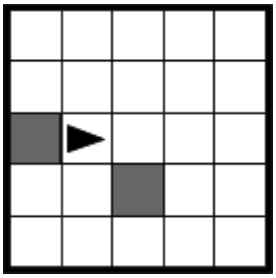




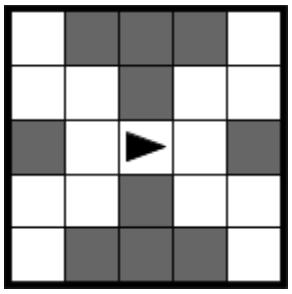
```
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
   IF (CAN_MOVE (left))
   {
      ROTATE_LEFT ()
      IF (CAN_MOVE (right))
      {
         ROTATE_RIGHT()
      }
   }
}
MOVE_FORWARD ()
```

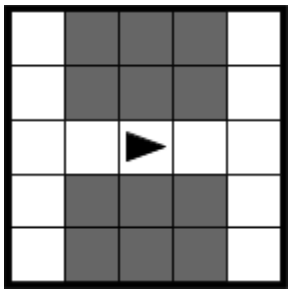


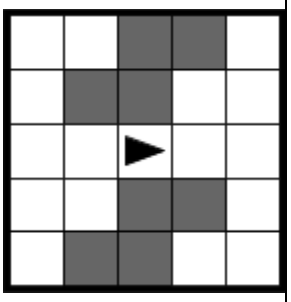




```
IF (CAN_MOVE (forward))
{
   MOVE_FORWARD ()
   IF( CAN_MOVE (left))
   {
      IF( CAN_MOVE (right))
      {
         ROTATE_RIGHT ()
      }
      ELSE
      {
         ROTATE_LEFT ()
      }
   }
   ELSE
   {
      ROTATE_RIGHT ()
   }
   MOVE_FORWARD ()
}
ELSE
{
   ROTATE_LEFT ()
   ROTATE_LEFT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





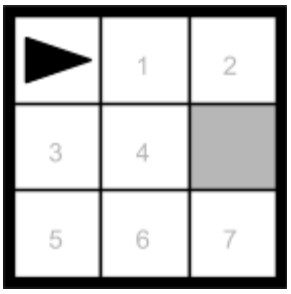


Now you try it!

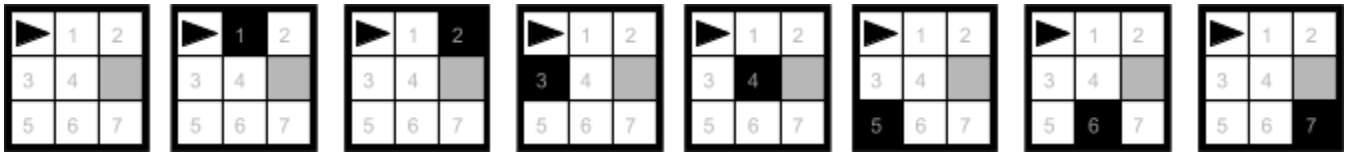
Now that you've had a bunch of practice reading and tracing code with if-statements, try writing your own pseudocode robot program that uses if-statements.

Problem statement:

Write a program to make the robot end up on the target gray square facing any direction...**but** your code must be able to handle the possibility of an obstacle that could appear in any one of the other squares (i.e. squares that aren't the start or target squares - numbered 1-7 in the diagram)



You must write the code without knowing ahead of time where the obstacle will be. In other words, you must *write one program* that can handle any possibility that might occur. For this exercise, there 8 possible locations where obstacle might be, we'll call them scenarios 0-7:



Write your program by hand below and test it by tracing it against each of the 8 possible scenarios. Your code should get the robot to the target no matter where the obstacle appears.
Goal: When the program ends, the robot is on the gray square -- it can be facing any direction
Tip: *When hand-writing code you don't need to follow the pseudocode syntax strictly, as long as your intent is clear. For example, using abbreviations and/or omitting the curly-braces and just indenting is fine.*