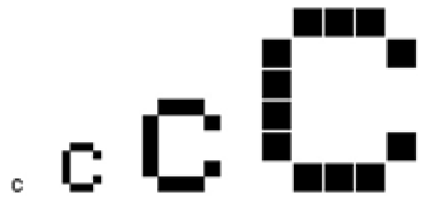
M286 Programming: Phase II	Teacher: Block:
Tech Tips: ● To print, click File>Print ● To edit, click the SIGN IN button in the upper, right corner. Then click File>Make a copy ● To download, click File>Download	
Course Standard & Goal	**Resources & Activities**
Curriculum Standard *Learning Goal (Parent-friendly language)*	❑ **Unit 1: Image Representation** – Use this resource to review how images are represented in computer language.
Curriculum Standard *Learning Goal (Parent-friendly language)*	❑ **Unit 2: Robots and Pseudocode** – Use this resource to learn how robots/turtles relate to pseudocode and how to use that in computer programming.
Curriculum Standard *Learning Goal (Parent-friendly language)*	❑ **Unit 3: Robot Programming** – More robots! Use what you have learned about robots and pseudocode to complete this activity.
❑ ***Must be turned in for course credit.**	

Unit 1: Image Representation

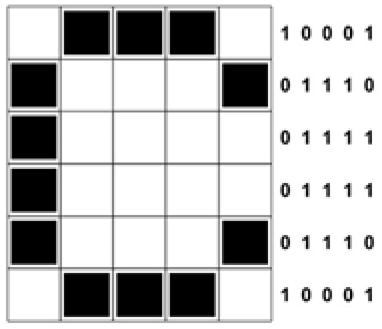
Everything you see on computational device screens (images and text) are images that a digital device has been programmed to display. Because computers store data as digits, computer images are ultimately represented inside a computer using binary 0's and 1's.

Pixels



In a black and white image, each pixel/dot can be either black or white, so all the computer would need to store is which dots are black and which are white. If we wanted to display the letter **C**, we first need to divide the letter into squares.

Binary Digits (bits)



If a 1 indicates 1 binary digit white square (1 bit) and a 0 indicates a black square then we can represent our letter C, on a 5x6 pixel grid, like this:

```
10001, 01110, 01111, 01111, 01110, 10001
```

Black & White (binary values) Activity

Directions: On the next page, color in each square with a 0 in it black, and leave each square with a 1 blank so that it is white, because 1 indicates that a pixel is 'on' (white) and a 0 indicates it is 'off' (and so it is black).

On the next page coloring in the squares makes a picture of a:

____

1	1	1	1	0	1	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	0	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1	0	0	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	0	0	0	1	1
1	0	0	1	1	1	0	1	1	1	1	1	0	0	0
1	1	0	1	1	0	0	1	1	1	1	1	0	0	0
1	0	0	1	1	1	0	1	1	1	1	1	0	1	1
0	0	1	1	0	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	1	1	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	0	1	0	1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	1	1	1	1	1	0	0	0
1	1	1	1	0	0	1	1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1	1	1	1	1	0	0	0
1	1	1	1	1	0	1	1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Black & White (2 binary values) in Run Length Encoding Activity

For the simplest of images, computers can use a compression algorithm called **run-length encoding (RLE).** Imagine that you had to read the bits above out to someone who was copying them down. After a while, you might say things like "five zeroes" instead of "zero zero zero zero zero".

In run-length encoding, the computer replaces each row with numbers that say how many consecutive pixels are the same color, always starting with the number of white or black pixels. For example, if the first row contains 3 white pixels, 2 black pixels, 5 white pixels, 2 black pixels, then 4 white pixels:

```
0001100000110000
```

This would be represented as follows: 3,2,5,2,4

Testing Your Knowledge 1

For the following row of the bitmap for the white and black values:

0011110001111000

How would this row be represented in RLE? **(Choose 1 answer)**

a. **0, 2, 5, 4, 5, 3**
b. **0, 2, 4, 3, 4, 3**
c. **0, 2, 5, 3, 5, 3**
d. **2, 4, 3, 4, 3**
e. **0, 2, 5, 3, 5, 3**

Directions: Color in each square with black or white values using the RLE values below for the grid on the next page. In this RLE first number always indicates a how many black values then switches back to the number of white values.

0, 15 (means 0 black vales and 15 values white)
0, 7, 1, 7 (0 black, 7 squares are white, 1 square is black, then 7 squares white)
0, 6, 1, 1, 1, 6 (0 black, 6 squares white, 1 black, 1 white, 1 black, 6 white)
0, 6, 1, 1, 1, 6
0, 5, 1, 3, 1, 5
0, 5, 1, 3, 1, 5
5, 5, 5 (5 squares black, 5 squares white, 5 black)
1, 13, 1
0, 1, 1, 3, 1, 3, 1, 3, 1, 1
0, 2, 1, 9, 1, 2
0, 3, 1, 1, 1, 3, 1, 1, 1, 3
0, 3, 1, 2, 3, 2, 1, 3
0, 2, 1, 9, 1, 2
0, 2, 1, 9, 1, 2
0, 1, 1, 5, 1, 5, 1, 1
0, 1, 1, 3, 2, 1, 2, 3, 1, 1
1, 2, 2, 5, 2, 2, 1
3, 9, 3
0, 15
0, 15

Filling in the Grid

Coloring in these squares makes a picture of a: ______________________________

Testing Your Knowledge 2

The following is a compression of a 6x6 black and white icon, using RLE:

```
2,2,2
2,2,2
0,6
0,6
2,2,2
2,2,2
```

What mathematical symbol does that icon resemble? **(Choose 1 answer)**

- **A.** /
- **B.** +
- **C.** >
- **D.** ×
- **E.** -

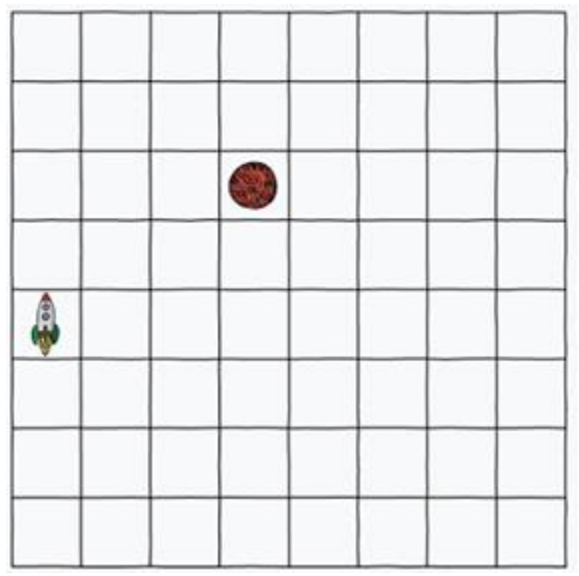
End of Unit 1: Image Representation

Unit 2: Robots and Pseudocode

Programming involves planning what you're going to do, "coding" the instructions, testing them, tracking down any bugs, and changing the program so it works correctly.

In these activities we will use a simple programming language called **Robot Code** or **Turtle Code** by creating directions of how to move along a grid or graph.

Functions (verbs)



In order to direct the rocket/robot to reach the planet we are going to create directions or code, not just remote controlling it, because ALL the instructions are written before the robot can follow those instructions.
In order to create clear instructions for the robot, verbs or **functions** (also called methods) need to be set for consistent directions.

Directions: in the below spaces use arrows to create all possible functions to instruct the rocket to the planet. Choose arrows to represent move forward, turn left and turn right.

Note: There is NO backwards for rockets!

↑				

What if there was only enough computer memory or storage space for 2 commands? How would you turn left AND right?

____

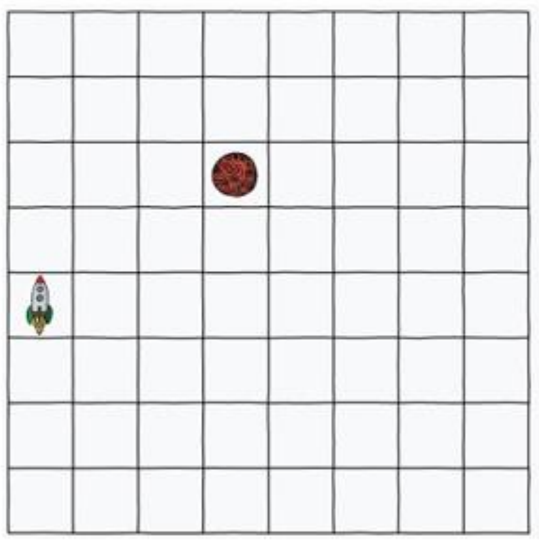
____

Now we are ready to program our **algorithm**, or list of instructions used to perform a task.

Directions: On the next page program your algorithm in order to get the robot to the planet.

Algorithm Exercise Phase I

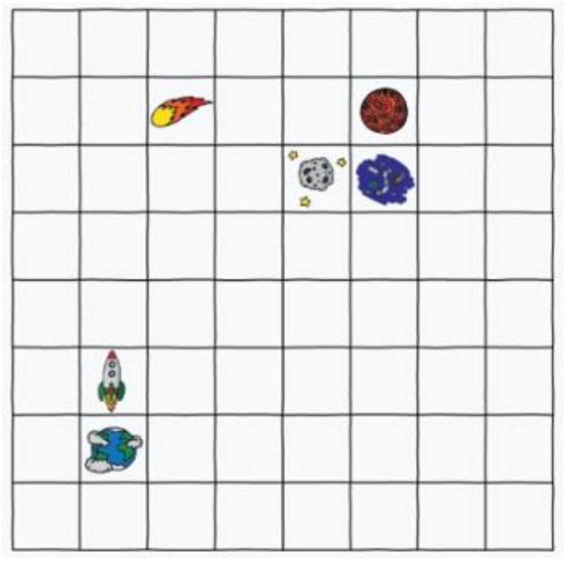
1. ☐
2. ☐
3. ☐
4. ☐
5. ☐
6. ☐



Application of Algorithm Activity

Barriers have been added to the grid so that the path is more complex because the robot needs to avoid the barriers. This could be space junk and comets, or you could invent a new scenario for the grid.

Directions: Program the trip without using the left hand turn (i.e. the only instructions are Forward and Right turn.) Try to keep the code <15 codes.
Hint: A left hand turn can be achieved by doing three Right turn instructions.
Hint: Depending on your path, you may or may not use all boxes here.



☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐

Question from the grid on the previous page: What is the minimal amount of additional barriers could you add to the grid on the previous page so the algorithm for the robot reaching the planet is IMPOSSIBLE?

____

Pseudocode

What if we wanted to program these directions into a computer but we did not know which programming language to use? We would construct our robot code in **pseudocode**, planning algorithms with sketching out the structure of the program before the actual coding takes place.

Pseudocode is understood by the programmers of all types, and can happen as block-based or text-based. In this example we will concentrate on text-based pseudocode.

In text-based pseudocode functions (verbs) have to be differentiated from nouns (also called data or variables). For example, the word move can be a verb and a noun.

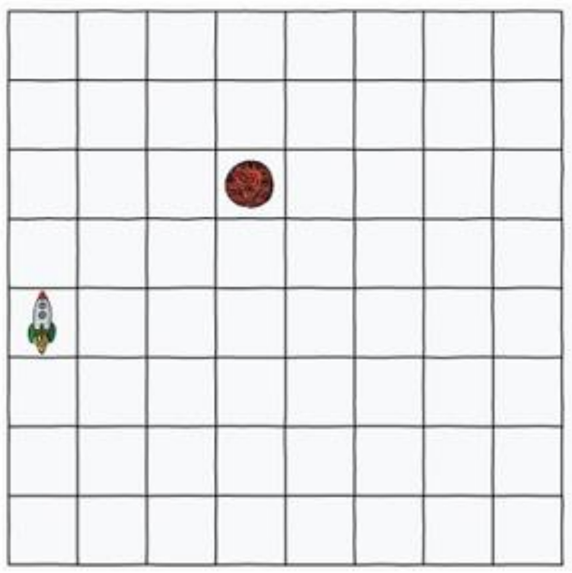
In pseudocode all functions are followed by open and closed parentheses to denote a function. So instead of this symbol ⇧ we would create

`move_forward()`

Directions: Use the robot code you created for Algorithm Phase I and create pseudocode.

Algorithm Exercise Phase II

1. ☐ `move_forward()`
2. ☐ ____________________
3. ☐ ____________________
4. ☐ ____________________
5. ☐ ____________________
6. ☐ ____________________

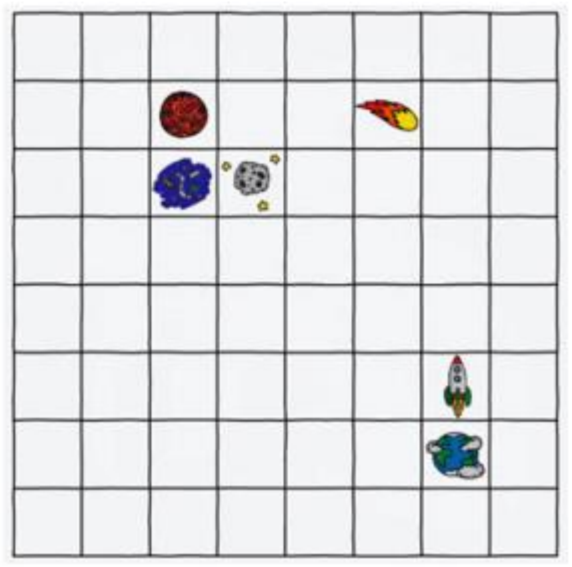


Debugging Pseudocode

Sometimes pseudocode has "bugs" in the program. A bug is when my program isn't doing what was expected.

Directions: On the left draw a circle around the pieces of code where you notice the instructions seem to be going wrong. On the grid below rewrite the code.

```
1. move_forward()
2. move_forward()
3. move_forward()
4. move_forward()
5. move_right()
6. move_right()
7. move_forward()
8. move_forward()
9. move_forward()
10.move_forward()
11.move_right()
```



Re-writing the Pseudocode

1.	11.
2.	12.
3.	13.
4.	14.
5.	15.
6.	16.
7.	17.
8.	18.
9.	19.
10.	20.

End of Unit 2: Robots and Pseudocode

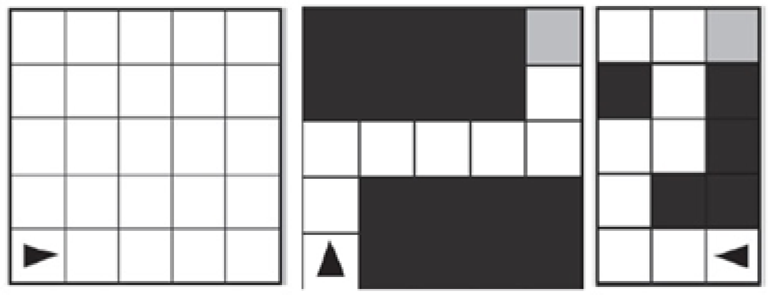
Unit 3: Robot Programming

Programming enables problem solving, human expression, and creation of knowledge.

There are typically two styles of pseudocode and Robot Pseudocode:

TEXT-BASED and BLOCK-BASED.

The key difference with the other questions is the output appearance. The robot questions involve a grid with a robot busily moving around the grid. Sample Grids are shown below:



Most grids will use a matrix of five rows and five columns, but grids can differ.

The robot is represented by a black triangle/turtle which can face in all four directions: up, down, left, and right.

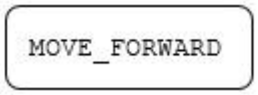
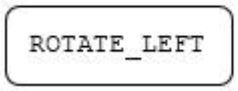
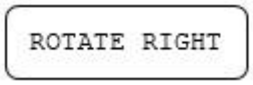
If the grid is not empty, some squares will be shaded gray (for destination/end position), but black squares indicate obstacles where the robot cannot enter. These packet questions will be in 2 categories:

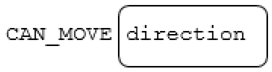
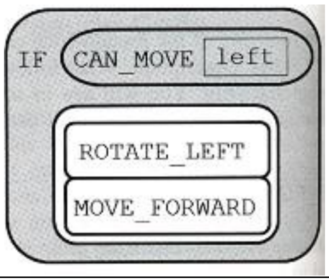
1. You are provided with a grid and a robot in a starting position. You are also provided with pseudocode (TEXT-BASED or BLOCK-BASED) and you must then select from four different grids that show the final position of the robot.

2. You are provided with a grid and a robot in a starting position and the robot needs to end up in a gray square (destination). You need to select from the given pseudocode (TEXT-BASED or BLOCK-BASED) that will achieve the desired goal.

Robot Pseudocode Functions (verbs)

The robot uses a combinations of pseudo-code commands or functions:

TEXT- BASED	BLOCK- BASED	DESCRIPTION
Text: MOVE_FORWARD()	**Block:**  MOVE_FORWARD	The robot moves one square forward in the direction it is facing.
Text: ROTATE_LEFT()	**Block:**  ROTATE_LEFT	The robot rotates in place 90 degrees counterclockwise (i.e., makes an in-place left turn).
Text: ROTATE_RIGHT()	**Block:**  ROTATE RIGHT	The robot rotates in place 90 degrees clockwise (i.e., makes an in-place right turn).

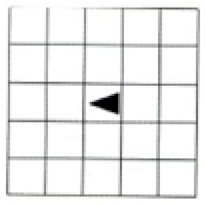
Text:
 	
CAN_MOVE (direction)
 	
or
 	
IF (CAN_MOVE (left))
{
ROTATE_LEFT()
MOVE_FORWARD()
}
	Block:
	
	CAN_MOVE direction
	
	IF CAN_MOVE left
	ROTATE_LEFT
	MOVE_FORWARD
		Evaluates to **true** if there is an open square one square in the direction relative to where the robot is facing; otherwise evaluates to **false**.
		The value of direction can be (left, right, forward, or backward).

Text:
 	
REPEAT UNTIL (GoalReached())
 	
{IF (CAN_MOVE(forward))
 	
    {
 	
    MOVE_FORWARD()
    }
}
		The GOAL_REACHED command is very practical for robot programs. The robot does not always move 20 times or 10 times, but makes moves until the job is done. In most cases this means that the robot has entered the gray (destination) cell.

Text: MoveAndTurn()		For MoveAndTurn, the actual procedure logic is shown by providing the program code of the procedure. This means that you must first comprehend what the intent of the procedure is and then apply the command logic to the question.

Robot Programming Exercises

Question 1: **(Robot Pseudocode)** The below figure shows a robot in a grid of squares. The black triangle represents the robot, which is initially facing left.

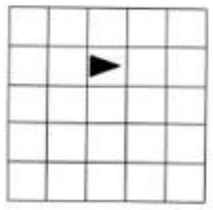


Consider the following code segment.

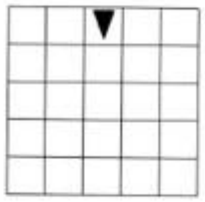
```
REPEAT 3 TIMES
{
   REPEAT 2 TIMES
   {
     MOVE_FORWARD ()
   }
   ROTATE_RIGHT ()
}
```

Which one of the four grids below shows the correct robot location after running the code segment? Circle the correct one.

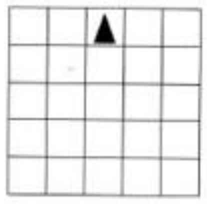
a.



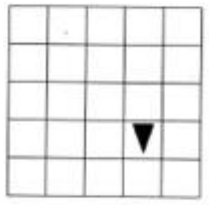
b.



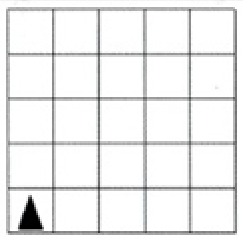
c.

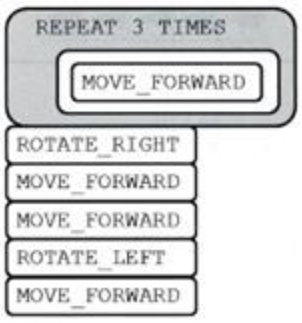


d.



Question 2: **(Robot Pseudocode)** The below figure shows a robot in a grid of squares.
The black triangle represents the robot, which is initially facing upward.



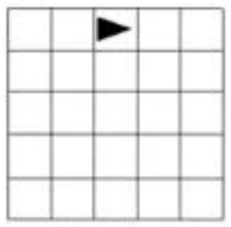


```
REPEAT 3 TIMES
    MOVE_FORWARD
ROTATE_RIGHT
MOVE_FORWARD
MOVE_FORWARD
ROTATE_LEFT
MOVE_FORWARD
```

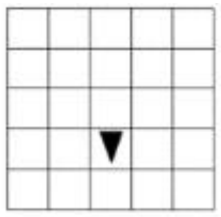
Consider the following block bode segment.

Which one of the four grids below shows the correct robot location after running the code segment? Circle the correct one.

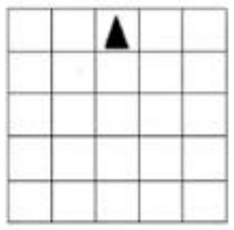
a.



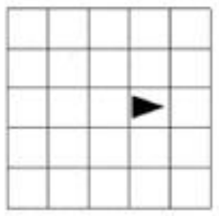
b.



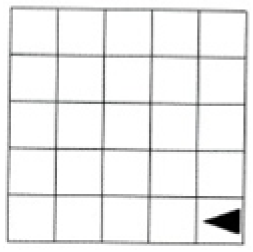
c.



d.



Question 3: **(Robot Pseudocode)** The below figure shows a robot in a grid of squares. The black triangle represents the robot, which is initially facing left.

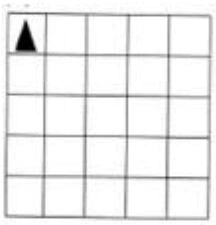


Consider the following code segment.

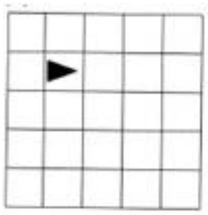
```
REPEAT 2 TIMES
{
   REPEAT 2 TIMES
   {
     MOVE_FORWARD ()
     ROTATE_RIGHT ()
     MOVE_FORWARD ()
     ROTATE_LEFT ()
}
```

Which one of the four grids below shows the correct robot location after running the code segment? Circle the correct one on the next page.

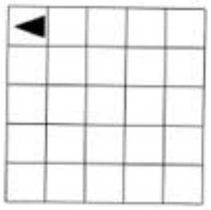
a.



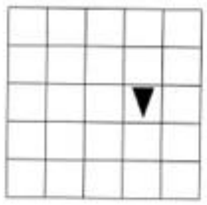
b.



c.

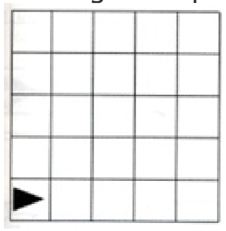


d.

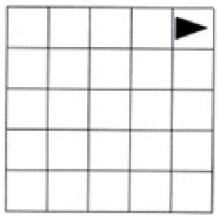


Question 4: **(Robot Pseudocode)** The two figures below each show a robot in a grid of squares. The black triangle represents the robot, which is initially facing right.

This figure represents the robot location at the start of the program execution.



This figure represents the location at the end of the program execution.



Which one of the following code segments will move the robot to the required square? Circle the correct code segment.

a.
```
REPEAT 4 TIMES
{
    MOVE_FORWARD ()
    ROTATE_RIGHT ()
    MOVE_FORWARD ()
    ROTATE_LEFT ()
}
```

b.
```
REPEAT 4 TIMES
{
    MOVE_FORWARD ()
    ROTATE_LEFT ()
    MOVE_FORWARD ()
    ROTATE_RIGHT ()
}
```

c.
```
REPEAT 4 TIMES
{
    MOVE_FORWARD ()
    MOVE_FORWARD ()
    ROTATE_LEFT ()
    ROTATE_RIGHT ()
}
```

d.
```
REPEAT 4 TIMES
{
    MOVE_FORWARD ()
    ROTATE_LEFT ()
    MOVE_FORWARD ()
}
```

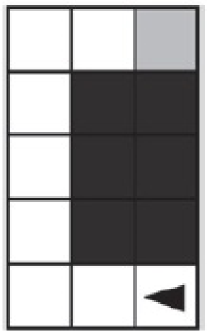
Question 5: **(Robot Pseudocode)** The below figure shows a robot in a grid of squares.

The program segment below is intended to move a robot in a grid to a gray square.
The program segment uses the procedure **`GoalReached()`**, which evaluates to **true** if the robot is in the gray square and evaluates to **false** otherwise.
The robot in each grid is represented as a triangle and is initially facing left.
The robot can move into a white or gray square but cannot move into a black region.

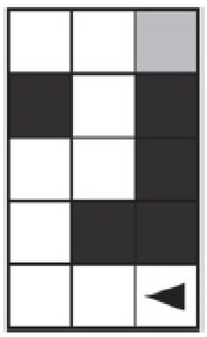
```
REPEAT UNTIL (GoalReached ())
{
   IF (CAN_MOVE (forward))
   {
     MOVE FORWARD ()
   }
   IF (CAN_MOVE (right))
   {
     ROTATE RIGHT ()
   }
   IF (CAN_MOVE (left))
   {
     ROTATE LEFT ()
   }
}
```

For which of the following grids does the program NOT correctly move the robot to the gray square?

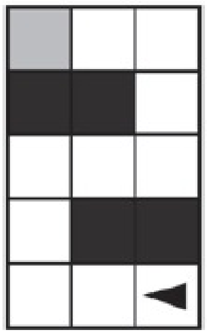
a.



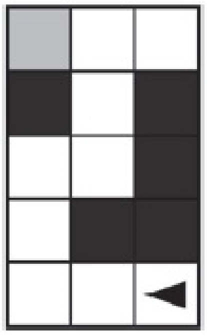
b.



c.



d.



End of Unit 3: Robot Programming