Robotics Badges: ***Programming Robots 2***

Robot Maze 2

THINGS TO KNOW:

Write computer programs: When programming with a computer language, it is important to get the syntax exactly right. **Syntax** is the set of rules used by a language. It includes spelling, spacing, symbols, and punctuation.

To plan a program without worrying about getting the syntax of a computer language exactly right, programmers sometime use **pseudocode**. ("Pseudo" means "fake.") Pseudocode uses commands in regular human language. It can then be translated into any kind of computer language.

The only rule for pseudocode is that it should be easy to understand.

For example, to write a pseudocode program to tell a robot how to walk from the kitchen to the bedroom, you might say:

1. FORWARD 20 steps
2. LEFT
3. FORWARD 10 steps
4. RIGHT
5. UP 15 steps
6. FORWARD 1 step
7. RIGHT
8. FORWARD 10 steps

Advanced programming concepts: Algorithms also contain rules that tell the computer what action to take when it has to make a decision. These rules are written in the form of conditional statements. They state the conditions that much exist for a computer to take an action.

Conditional statements can be written as IF-THEN or IF-THEN-ELSE. For example, to tell your lawn-mowing robot to go around a rock, you could write "IF meet an obstacle, THEN go around it. ELSE continue straight."

You can use shortcuts to avoid writing the same set of commands over and over. For example:

- A **loop** tells the program to go back and repeat a series of commands.
- A **function** is a series of commands that is given a name. When you "call" the function, those steps are carried out.

(continued)

For loops and functions, it helps to indent the steps inside them. Here are some examples of commands you can use for your pseudocode:

COMMAND	EXAMPLE
For decisions, use conditional statements. These are written in the form of IF-THEN or IF-THEN-ELSE.	IF Light = on THEN go forward 1 step ELSE stop.
For loops, use beginning and ending statements. To start the loop, write REPEAT and then how long the loop should keep repeating, such as REPEAT FOREVER or REPEAT [number] TIMES. You can also control the loop with a conditional statement. REPEAT WHILE needs to meet the condition to start the loop. REPEAT UNTIL will stop the loop when the condition is met.	REPEAT UNTIL Light = off Move ahead 1 step Turn left Move ahead 2 steps END
For functions, you need to define the function. Then you can call the function by inserting the name where you want those steps to run. Here's an example of a function called "cross" that tells the robot how to cross the street. (Notice that the function contains a REPEAT UNTIL loop.) Whenever you want to use those exact steps, you can write "CALL cross" instead.	FUNCTION cross REPEAT UNTIL traffic = none Look left Look right END FORWARD 50 steps END FUNCTION

(continued)

First, use the "Left Hand on Wall" method that is often used for maze-solving robots and translate it into pseudocode. The "Left Hand on Wall" rule is based on the idea that if you are walking through a maze and always keep the same hand on the wall, eventually you will end up at the exit.

Here is what the rule says:

- Always turn left if you can.
- When you can't turn left, go straight.
- When you can't turn left or go straight, turn right.
- When you can't turn left, go straight, or turn right, turn around because you must be at a dead end.

LEFT HAND ON WALL IN PSEUDOCODE:

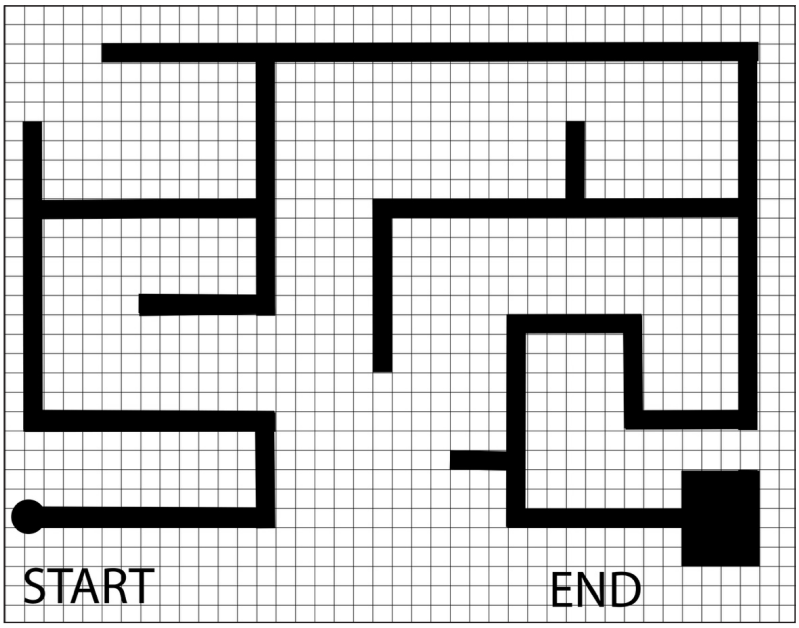
(continued)

Now, write a program using loops and conditional statements that will take a robot from the beginning to the end of any simple maze. The robot may take wrong turns, so it won't be as efficient as when you just gave it "driving directions."

PROGRAM:

(continued)

Finally, test your program with the below maze to see if it program works. If there is a **bug** or mistake in the program, try to fix it!



Want a Challenge? Try this!

Robots can run more than one program at the same time. For example, while the maze-solving program is running, the robot could also be running code that tells it to follow a line.

Now that the programs tell the robot how to solve a maze, challenge teams to write pseudocode with conditional statements, loops, and functions to create a program that makes a robot follow the turns of a line.

To help, here's how line-following robots work:

- A line-following robot usually has two light sensors, one on each side, that control the wheels on that side.
- The light sensors can detect the difference between the dark line and the white paper.
- As long as the light sensor detects light (the white paper), the wheel it controls will keep turning. So as long as the black line is centered under the robot, the sensors on the sides will detect white paper. With both wheels turning at the same speed, the robot will move forward.
- When the light sensor detects dark (the black line) on one side, the wheel on that side stops. Meanwhile, the other wheel, which is still over the white paper, keeps going. This causes the robot to pivot around in the direction of the black line.
- When the robot has turned enough so that the black line is centered beneath it, the second wheel begins to turn again, and the robot moves forward.
- When both sensors detect dark (the black box at the end of the maze), both wheels stop turning and the robot stops moving.