Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Materials***

The third mission is to program your robot to navigate a maze, retrieve a secret package, and return to the original starting point. As always should your robot be discovered or captured, your teacher will disavow any knowledge of your mission. Good luck.

You need:

- ❏ 1 Norland Calculator Robot
- ❏ 1 Graphing Calculator
- ❏ Several Meter Sticks
- ❏ Graph Paper

Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Instructions***

Discuss how many different ways you can program your robot to turn. Which ways might be best for navigating a maze? How can you make a 90˚ turn? Layout a practice maze with meter sticks on each side about one foot apart. Start with two straight runs with a right angle turn in between them. Create the new program **MAZE** (see PROGRAMMING INSTRUCTIONS if needed). When programming your robot, recall the following numbers used in a Send command. For example, Send ({ABC,xxx}):

A-Time or Bumper	**B-Left Wheel**	**C-Right Wheel**
1=timed movement only	0=backwards	0=backwards
2=move until bumper hits	1=no motion	1=no motion
3=time or until bumper hits	2=forwards	2=forwards

xxx is the number of seconds of run time in centiseconds.

For example: Send ({122,600})
Get (R) (Always needed to close a Send command.)

The robot will move forward for 6 seconds.

You'll need to know how fast your robot travels. For example, if your robot takes 5.27 seconds to travel the distance of one meter stick or 100 cm, it's traveling at approximately 18.98 cm per second (r=d/t or r=100/5.27).

When you've discovered how to make your robot rotate for a turn, you'll need to determine how many seconds the rotation must last for a 90 degree turn.

Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Challenge***

The official test maze will have four straight runs and three turns. At the end of the maze there will be a secret message cube that must be recovered. Attach something to the robot so that the message cube can be retrieved.

The chart that follows can help you plan your strategy for completing the maze.

Maze Chart

Run 1 (In centimeters)	Time Needed (In centiseconds)	Commands
Turn 1 (Left or Right)		
Run 2 (In centimeters)		
Turn 2 (Left or Right)		
Run 3 (In centimeters)		
Turn 3 (Left or Right)		
Run 4 (In centimeters)		

Grading Scale:

Robot retrieves message cube and returns it to start: **A+**
Robot retrieves message cube and spins in circle for joy: **A**
Robot makes it through, but misses message cube: **B**
Robot makes it halfway through the maze: **C**

Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Results***

1. List five different environments where it might be too dangerous for humans to explore, but a robot could go and send back valuable information.

1.

2.

3.

4.

5.

2. Describe three situations where humans couldn't reach and you would need a small robot to explore.

1.

2.

3.

3. Have you seen the movie, *Fantastic Voyage*? Could miniature robots be used to explore the human body? Draw a picture of miniature robot below and explain what devices it might have to explore the human body?

Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Extension***

1. Place a small piece of tape labeled Point A on the floor. Place another piece of tape five feet (feet: primitive units of measure) away and label it Point B. Program your robot to get from Point A to Point B, but your journey must include one right angle (90°) turn at a point we'll call Point C. Measure the distances your robot travels before and after Point C. Below, draw the right triangle formed by points A, B, and C. Label the distances between each point to the nearest whole foot.

2. If points A and B in the description above were 10 feet apart, what would the lengths of the other segments be? Draw and label the new triangle formed.

3. If Point A and B were 13 feet apart, what would be the shortest distance your robot would travel before making a 90° turn at Point C?

Name:			Date:
MISSION	**3**	**Turns and Mazes**	***Programming Instructions***

Turn on your graphing calculator. Press [PRGM] and arrow to highlight **NEW**. Press [ENTER], then spell out [MAZE] by pressing the appropriate keys. Press [ENTER] and you're ready to enter the first command for the program.

Line 1: Press [PRGM], then use arrow to highlight **I/O**. Use arrow to scroll down to **B: Send (** and press [ENTER]. Press [2nd] and then press [(] for an open brace. Type in [1][2][2][,][6][0][0]. Close the braces and parentheses by pressing [2nd], [}], then [)]. Press [ENTER]. The first line should appear as:
:Send({122,600})

(**Line 2:** Is blank)

Line 3: Press [PRGM], then arrow to highlight **I/O**. Use the arrow to scroll down to **A: Get (**, and press [ENTER]. Press [ALPHA], then [R]. Press [(] then [ENTER]. The third line should appear as:
:Get(R)

Line 4: Press [PRGM], then use arrow to highlight **I/O**. Use arrow to scroll down to **B: Send (** and press [ENTER]. Press [2nd] and then press [{]. Type in [1][2][0][,][4][2]. Close the braces and parentheses by pressing [2nd], [}], then [)]. Press [ENTER]. The fourth line should appear as:
:Send({120,42})

Line 5: Press [PRGM], then arrow to highlight **I/O**. Use the arrow to scroll down to **A: Get (**, and press [ENTER]. Press [ALPHA], then [R]. Press [)] then [ENTER]. The fifth line should appear as:
:Get(R)

Line 6: Press [PRGM], then use arrow to highlight **I/O**. Use arrow to scroll down to **B: Send (** and press [ENTER]. Press [2nd] and then press [{]. Type in [1][2][2][,][6][0][0]. Close the braces and parentheses by pressing [2nd], [}], then [)]. Press [ENTER]. The sixth line should appear as:
:Send({122,600})

(**Line 7:** Is blank)

Line 8: Press [PRGM], then arrow to highlight **I/O**. Use the arrow to scroll down to **A: Get (**, and press [ENTER]. Press [ALPHA], then [R]. Press [)] then [ENTER]. The eighth line should appear as:
:Get(R)

Adjust command times as necessary. Add forward motion and turns as needed. A sample left turn is, :Send({102,42}).

There are several ways to make the robot turn: one wheel stopped and the other moving forward or backward, one wheel moving forward and the other moving backward, et cetera. A sample right turn would be: Send ({120,42}) followed by Get (R). Hopefully by now most students are becoming comfortable programming robot movements on the calculator. However, if needed, a starter program that includes two straight runs with a right angle turn in between can be found in the PROGRAMMING INSTRUCTIONS. An alternative programming method is to use two programs and the recall command.

For example, first have students experiment with programming turns in their **GO** program from **Mission 1**, then have them program a straight run and a right turn. They can then create the new program **MAZE** and repeatedly recall sets of instructions from the **GO** program as follows.

From the new **MAZE** program press [2nd], then [RCL] . Press PRGM, then scroll over to **EXEC** and select **GO**. Press [ENTER] twice and the instructions from **GO** will be added to **MAZE**. Edit commands and times as necessary. This process can be repeated as many times as needed.

Sometimes just getting through the maze is challenging enough. One way to create a maze is to use meter sticks about a foot apart and included left and right turns. Place a paper cube (templates available on Internet) with a message inside at the end. A rolled piece of tape on the robot's bumper works for "picking up" the cube or Velcro strips or dots can be used.

For questions 1-3 answers will vary. The extension activity involves right triangles, the Pythagorean Theorem, and Pythagorean Triples. For **question 1**, the sides should be labeled 3 and 4 with a hypotenuse of 5. For **question 2**, the sides are 6 and 8. For **question 3**, the answer is 5. A linoleum floor composed of one-foot square tiles is helpful to visualize the right triangles formed by the robots' movements.

Another extension is to have students use their maze programming skills and have their robots duplicate the first iteration of the Jurassic Park fractal. See: *http://math.rice.edu/~lanius/frac/real.html*. *

* Used with permission of Cynthia Lanius