# Recursion

An example of a recursive method is shown in the example file for setting the lights. Unlike the previous example for setting a 2D array of lights, the recursive version uses no for loop. It recursively calls itself while the column value is less than the limit. In the recursiveLights() method, the limit is known to be 8, however, it could be dynamically passed as a parameter or set as an instance variable.

Each time the column value is less than the limit, the current column index values are used with the setLight() method and the column value updated. The method calls itself traveling deeper in the stack of method calls. Once the column value is larger enough, the method stops calling itself and instead calls the syncLights() method, each method from the deepest to the first call at the top level is able to complete and the main program continue.

1. **Based on the description and logic, how many calls deep does the recursiveLights() method go to set the lights?**

The second recursive method involves distance data readings and multiple conditions.

1. **Theorize what would happen if the distance threshold was not quickly crossed and there was no number of readings threshold.**

Typically, recursion should be implemented if it has a significant impact on simplifying the code. However, it should be avoided if larger local arrays are involved as it can result in a memory overflow if there are too many recursive calls.

Recursive methods can be used to hide a private helper method when used with a public driver method. In cases like these, the public method would call the first case of the private method, hiding the method from the user and potentially increasing the efficiency of the program. With respect to the efficiency of the program, the driver method can perform checks on possible erroneous input values before calling the recursive method.

We can create a simple driver and recursive helper method that will display a message along with a countdown.

1. **Create a public driver method in the RecursiveDistance class that takes an input of count value and only runs the recursive method if the count is less than 15. A message telling the user the count is too large should display when the passed count value is greater than or equal to 15.**
2. **Add a private method called countdown() which gets called by the public driver method with its passed in count value. Add logic to display to the console the message “ seconds left” with the current count preceding it and to decrement the count variable. The method should pause 1 second each time and stop displaying messages after displaying a count value of 0 in the message. Hint: you can wait with the GroundRobot method waitTime() which takes as input milliseconds.**

Make sure to test your answers as you write them to check if they are correct.