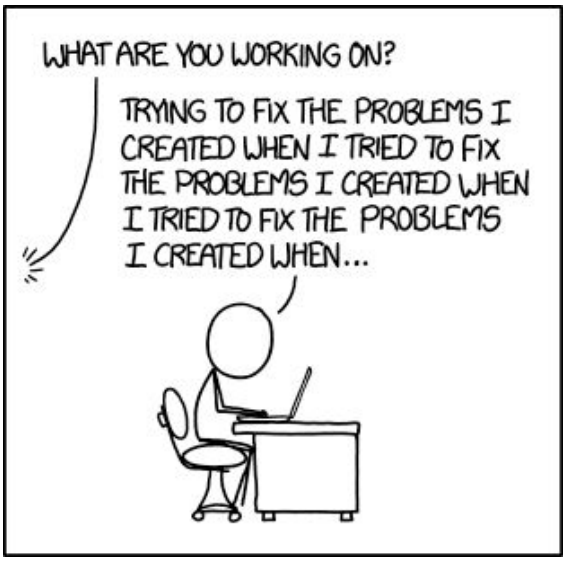
1-D Search - Two Directions



Our goal is to search for objects in 2-D space - in order to do that, we need to search in 1 dimension first using recursion. This challenge is different, non-trivial, and more applicable for recursion than the first 1-D search challenge.

There are three main differences from the first 1-D search function:

1. The length of the array MUST be an odd number. You must use input verification to ensure this.

2. The beginning point of the recursive search is the middle point of the array, and the number of steps is from the midpoint to the randomly placed 'X').

3. There will be two recursive calls, neither associated with a "return" value, each that searches in one direction at a time.

Your recursion function will continue to have three variables - the array, the current position (which will increase by 1 each recurse) and a counter (which will also increase by 1 each recurse). The counter and current position may seem redundant, but it will become essential when searching in 2-D (or 3-D) space.

Examples:

```
Scan in a length: 21
**X******************

There are 8 steps from the midpoint to the 'X'.

Scan in length: 5
****X

There are two steps from the midpoint to the 'X'.
```