Merge Sort

Problem Domain

- Function that takes in an array as a parameter, sort by merging in place, and output will be sorted array

Input: array

[38, 27, 43, 3, 9, 82, 10]

Output: sorted array

[3, 9, 10, 27, 38, 43, 82]

[38, 27, 43, 3, 9, 82, 10]

Mid = Math.floor(arr.length / 2) = 3
Left = [38, 27, 43]
Right = [3, 9, 82, 10]

```
  [ 38, 27, 43 ]          [ 3, 9, 82, 10]

 [38]  [27, 43]             [ 3, 9 ]    [ 82, 10 ]

[ 38 ]     [ 27 ]  [ 43 ]       [ 3 ]   [ 9 ]    [ 82 ]  [ 10 ]
```

Algorithm

- Main mergeSort function
    - Find a middle point (Math.floor just in case odd number of items in the array)
    - “Break” the array at the middle point, use recursion to further break down each half to get to base case
    - Base case- array with one item, naturally already sorted
    - Call the helper merge function with left, right, input array
    - Return the sorted array
- Separate, helper function merge
    - Take in left, right, array as parameters
    - Set an index variable, this will be dynamic, will increase as we continue to evaluate the base cases
    - Evaluate the values if they’re higher or lower, and adjust the index accordingly

Pseudo Code

```
- Mergesort (arr)
    - If arr.length === 1, return input arr
    - If arr.length > 1, do the work
    - Mid = Math.floor(arr.length / 2)
    - mergeSort(left)
    - mergeSort(right)
    - merge(left, right, arr)
    - Return arr
- Merge (left, right, arr)
    - Let index = 0, increment
    - Evaluating arr[index], adjusting left and right accordingly
```

Big O Notation

- Space: O(n)
- Time: O(n log n)

Testing

- Edge case: array of 1 item, input array is reverse sorted, negative numbers, duplicates in the array