The MergeSort function repeatedly divides the array into two halves until we reach a stage where we try to perform MergeSort on a subarray of size 1 i.

After that, the merge function comes into play and combines the sorted arrays into larger arrays until the whole array is merged.To sort an entire array, we need to call `MergeSort(A, 0, length(A)-1)`.

```
MergeSort(A, p, r):

    if p > r

        return

    q = (p+r)/2

    mergeSort(A, p, q)

    mergeSort(A, q+1, r)

    merge(A, p, q, r)
```

Every recursive algorithm is dependent on a base case and the ability to combine the results from base cases. Merge sort is no different. The most important part of the merge sort algorithm is, you guessed it, `merge` step.

The merge step is the solution to the simple problem of merging two sorted lists(arrays) to build one large sorted list(array).

The algorithm maintains three pointers, one for each of the two arrays and one for maintaining the current index of the final sorted array.

A noticeable difference between the merging step we described above and the one we use for merge sort is that we only perform the merge function on consecutive sub-arrays.

This is why we only need the array, the first position, the last index of the first subarray(we can calculate the first index of the second subarray) and the last index of the second subarray.

Our task is to merge two subarrays `A[p..q]` and `A[q+1..r]` to create a sorted array `A[p..r]`. So the inputs to the function are A, p, q and r

The merge function works as follows:

1. Create copies of the subarrays `L <- A[p..q]` and `M <- A[q+1..r]`.
2. Create three pointers `i`, `j` and `k`
    a. `i` maintains current index of `L`, starting at 1
    b. `j` maintains current index of `M`, starting at 1
    c. `k` maintains the current index of `A[p..q]`, starting at `p`.
3. Until we reach the end of either `L` or `M`, pick the larger among the elements from `L` and `M` and place them in the correct position at `A[p..q]`
4. When we run out of elements in either `L` or `M`, pick up the remaining elements and put in `A[p..q]`