```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    // Best: O(nlog(n) time | O(n) space
    // Average: O(nlog(n) time | O(n) space
    // Worst: O(nlog(n) time | O(n) space
    func mergeSort(_ array: inout [Int]) -> [Int] {
        if array.count <= 1 {
            return array
        }

        var auxiliaryArray = array
        mergeSortHelper(0, array.count - 1, &array, &auxiliaryArray)

        return array
    }

    func mergeSortHelper(_ startIndex: Int, _ endIndex: Int, _ firstArray: inout [Int], _ secondArray: inout [Int]) {
        if startIndex == endIndex {
            return
        }

        let middleIndex = Int(Double((startIndex + endIndex) / 2).rounded(.down))

        mergeSortHelper(startIndex, middleIndex, &secondArray, &firstArray)
        mergeSortHelper(middleIndex + 1, endIndex, &secondArray, &firstArray)
        doMerge(startIndex, middleIndex, endIndex, &firstArray, &secondArray)
    }

    func doMerge(_ startIndex: Int, _ middleIndex: Int, _ endIndex: Int, _ firstArray: inout [Int], _ secondArray: inout [Int]) {
        var k = startIndex, i = startIndex, j = middleIndex + 1

        while i <= middleIndex, j <= endIndex {
            if secondArray[i] <= secondArray[j] {
                firstArray[k] = secondArray[i]
                i += 1
            } else {
                firstArray[k] = secondArray[j]
                j += 1
            }

            k += 1
        }

        while i <= middleIndex {
            firstArray[k] = secondArray[i]
            i += 1
            k += 1
        }

        while j <= endIndex {
            firstArray[k] = secondArray[j]
            j += 1
            k += 1
        }
    }
}

```