Exercise 1

Question 2

The code written has a time complexity of O(n log(n)). Within the code, the function 'merge_seperate' goes on to divide the array in half n times until all the elements have been separated. Then the 'merge' function compares the left and right side elements to one another and merges them into one subarray from lowest to highest. It'll repeat until there's multiple subarrays. Next, the subarrays are compared to each other and swapped around to create a bigger subarray until the full array is sorted.

Question 3

[8, 42, 25, 3, 3, 1, 27, 3]

- First 'merge_seperate' splits array into left array of [8, 42, 25, 3] and right array of [3, 1, 27, 3].
- Then the left array [8, 42, 25, 3] splits further into [8, 42] and [25, 3]
- [8, 42] splits into [8] and [42],
- [25, 3] splits into [25] and [3].

Once the left elements have been fully split, the function works on the right side.

- [3, 1, 27, 3] splits into [3, 1] and [27, 3].
- [3, 1] splits into [3] and [1]
- [27, 3] splits into [27] and [3].

Finally once all the elements have been divided, the code moves onto comparing and merging through the 'merge' function.

- The function compares [8] and [42] and arranges into [8, 42]
- Function compares [25] and [3] and arranges into [3, 25]
- Then [8, 42] and [3, 25] combines and arranges into [3, 8, 25, 42]
- [3] is compared to [1] and combined into [1, 3]
- [27] is compared to [3] and combined into [3, 27]
- [1, 3] is compared to [3, 27] and combined into [1, 3, 3, 27]
- Finally [3, 8, 25, 42] is compared to [1, 3, 3, 27] and arranged to make [1, 3, 3, 8, 25, 27, 42]

Question 4

The steps match with the complexity since the splitting process took n steps which was 7, while the merging also had a linear manner.