Problem 1  [DPV] Problem 2.23 Part (a) – Majority in $O(n \log n)$ time

Explain your algorithm in words and analyze its running time including stating the recurrence. You can give pseudocode to help clarify but it is not required.
Problem 2  Counting Inversions

You are given a permutation \( A = [a_1, a_2, \ldots, a_n] \) of \( \{1, \ldots, n\} \) where \( n \) is a power of 2. Thus, the array \( A \) contains each number between 1 and \( n \) exactly once. An inversion is a pair of indices \( i \) and \( j \) where \( i < j \) and \( a_i > a_j \). Our goal is to count the number of inversions in \( A \).

As an example, consider \( A = [2, 4, 3, 1, 5] \). This has 4 inversions: \((2, 1), (4, 3), (4, 1), (3, 1)\).

Explain your algorithms in words and analyze its running time including stating the recurrence where relevant. You can give pseudocode to help clarify but it is not required.

Part (a): Give an \( O(n^2) \) time algorithm to count the number of inversions between the left and right halves, i.e., the number of pairs \( i, j \) where \( 1 \leq i \leq n/2, n/2 < j \leq n \) and \( a_i > a_j \). This is iterative, not divide and conquer.

Part (b): Suppose the left half is sorted and the right half is sorted (but no relation between the halves). Now give an \( O(n) \) time algorithm to count the number of inversions between the left and right halves, i.e., the number of pairs \( i, j \) where \( 1 \leq i \leq n/2, n/2 < j \leq n \) and \( a_i > a_j \). This is also iterative, not divide and conquer.

Part (c): Give a divide-and-conquer algorithm which takes as input a permutation \( A \) and computes the number of inversions in \( A \). The running time should be \( O(n \log n) \).