Problem 1

Solution.
(b) The time complexity of the bubble sort is $O(1)$ times the maximum number of the iterations. So the time complexity should be $O(n^2)$
(c) The worst case happens when the largest number is $A[0]$. The time complexity for the worst case should be $O(n^2)$
(d) The best case happens when the array is given in ascending order. In the best case, the bubble sort need only to go through the array, check the swap condition as false and halt. The time complexity is $O(n)$

Problem 2

Solution.
(a) No. Because the Big-Oh describes the upper bound of the algorithm.
(b) No. Because the Big-Omega describes the lower bound of the algorithm.
(c) No. Because the notation only indicates the growth of the two algorithms is same. The exact time may vary.

Problem 3

Solution.
The mergesort works as follows:
1. If the list is of length 0 or 1, then it is already sorted. Otherwise:
2. Divide the unsorted list into two sublists of about half the size.
3. Sort each sublist recursively by re-applying the merge sort.
4. Merge the two sorted sublists back into one sorted list.

Problem 4

Pseudocode:
Insertion:
1. If treeNode is None, treeNode = NewNode
2. Else if (NewNode.key < TreeNode.key), Insert NewNode to left tree recursively.
3. Else, Insert NewNode to right tree recursively.