VE 281 Quiz 1

Name: , ID No.:

- **Problem 1.**

  We want to sort the following array in ascending order using quickSort (taught in class):

  7 2 8 5 10 6 9 4 1 3.

  We use the `quickSort(int A[], int left, int right)` function, where our "random" function always chooses **the first element** in array `A` as the pivot. Write down all the calls of `quickSort` until `A` is sorted. Hint: remember quickSort has two recursive calls: left than right. You may use the non-in-place partition for this question. Please annotate the version of the partition algorithm you pick.

  1st **call:** `quickSort({7, 2, 8, 5, 10, 6, 9, 4, 1, 3}, 0, 9)`
  2nd **call:** ...

- **Problem 2.**

  Write the in place partition in pseudo code or plain English. And illustrate the operations of partition on the arrary: 6, 2, 8, 5, 11, 10, 4, 1, 9, 7, 3 (with the first element 6 as the pivot) step by step using the algorithm.

- **Problem 3.**

  Is selection sort stable?

  Assume we have the following elements and we use the integer part of the element as the key in ascending selection sort.

  (2, a) (2, b) (3, a) (1, a)

  Assume we **always swap** the current element with the smallest remaining element in the unsorted part of the array, even if they have equal keys. Please arrange the above elements into an array such that the stability is violated (if there exists any).

  Assume we **never swap** the current element with the smallest remaining element if they have equal keys. Please arrange the above elements into an array such that the stability is violated (if there exists any).

  Is bubble sort stable? Explain why (both bubble sort and selection sort swap elements. Please focus on their similarity and differences).

- **Problem 4.**

  With regard to all the algorithms we talked about in class (the class versions), please fill in the following table (use the big-O with the tightest bounds covered in class):

	Worst-Case Time (Big-O)	Average-Case Time (Big-O)	In Place (Yes/No)	Stable (Yes/No)
Insertion Sort				
Selection Sort				
Bubble Sort				
Merge Sort				
Quick Sort				

- **Problem 5.**

  In class, in the last step of the counting sort algorithm, we iterate the original unsorted array `A` backward as we fill in the sorted array. Explain all the necessary updates needed for the algorithm in order to traverse the original unsorted array `A` in a forward fashion.

  Current steps in counting sort:

  1. Allocate an array `C[k+1]`.
  2. Scan `A`, `For i=1 to N: increment C[A[i]]`.
  3. `For i=1 to k: C[i] = C[i-1] + C[i]`.
  4. `For i=N to 1: put A[i] in new position C[A[i]], then decrement C[A[i]]`.