**LEBANESE AMERICAN UNIVERSITY**

**Department of Computer Science and Mathematics**

**CSC 310: Algorithms and Data Structures**

Fall 2015

**Lab III**

**Problem 1**

Implement selection sort algorithm which goes as follows: Finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

**Sample Input: Sample Output:**

3

**7** 25 13 10 30 15 27 37 10 13 15 25 27 30 37

**4** 6 7 8 9 6 7 8 9

**6** 10 7 15 13 4 6 4 6 7 10 13 15

**Problem 2**

Implement insertion sort which goes as follows: At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position..

**Sample Input: Sample Output:**

3

**7** 25 13 10 30 15 27 37 10 13 15 25 27 30 37

**4** 6 7 8 9 6 7 8 9

**6** 10 7 15 13 4 6 4 6 7 10 13 15

**Problem 3**

Implement bubble sort algorithm which goes as follows: Keep comparing pairs of adjacent elements of the sequence, if the they are in the wrong order swap them and do this till there are no swappings to do.

**Sample Input: Sample Output:**

3

**7** 25 13 10 30 15 27 37 10 13 15 25 27 30 37

**4** 6 7 8 9 6 7 8 9

**6** 10 7 15 13 4 6 4 6 7 10 13 15

**Problem 4**

Implement counting sort algorithm which goes as follows: loop over the items, computing a [histogram](https://en.wikipedia.org/wiki/Histogram) of the number of times each key occurs within the input collection. It then performs a [prefix sum](https://en.wikipedia.org/wiki/Prefix_sum) computation (a second loop, over the range of possible keys) to determine, for each key, the starting position in the output array of the items having that key. Finally, it loops over the items again, moving each item into its sorted position in the output array

**Sample Input: Sample Output:**

3

**7** 25 13 10 30 15 27 37 10 13 15 25 27 30 37

**4** 6 7 8 9 6 7 8 9

**6** 10 7 15 13 4 6 4 6 7 10 13 15