**LEBANESE AMERICAN UNIVERSITY**

**Department of Computer Science and Mathematics**

**CSC 310: Algorithms and Data Structures**

Fall 2013

**Lab IV**

**Complex Sorting**

**Input:**

All inputs are read from a file labeled “sort**N**.in” where **N** is the problem number. In the file, You read an integer **k** the number of test cases then **k test** cases follow. For each case, we read an integer **n** then **n** integers follow which specifies the number of elements in the array.

**Input:**

k

**n** n1 n2 n3…

…

**Output:**

Output should be consistent with the output specified in each problem as specified.

**Problem 1: [Radix Sort]**

Implement Radix sort.

**Sample Input: Sample Output:**

3

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

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

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

**Problem 2: [Count Sort]**

Implement count sort. The counting sort counts the number of occurrences of each element in an 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

**Problem 3: [Quick Sort]**

Implement Quick Sort.

**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: [Heap Sort]**

Implement Heap Sort.

**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 5: [Merge Sort]**

Implement Merge Sort.

**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 6: [Tree Sort]**

Implement Tree Sort. A binary search tree which is stored in an array instead of making a BST class and you print them in-order DFS.

**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