**Assignment 8**

**A. Which, if any, of the following algorithms, bubble-sort, heap-sort, insertion sort,  
merge-sort, and quick-sort, are stable? Briefly justify your answer.**Insertion sort, merge sort and quick sort are stable algorithm because these sort maintain the relative order of the elements in case they are already sorted.

**B. Is the bucket-sort algorithm in-place? Why or why not?**It is not an in-place algorithm because the elements are sent to a separate bucket while sorting.

**C. Illustrate the performance of the radix-sort algorithm on the following input sequence**(22, 15, 26, 44, 10, 3, 9, 13, 29, 25).

(22, 15, 26, 44, 10, 03, 09, 13, 29, 25).(10, 22, 03, 13, 44, 15, 25, 26, 09, 29). // arranged 0th digits first

(03, 09, 10, 13, 15, 22, 25, 26, 29, 44). // arranged 10th digits.

**D. Note that the Priority Queue ADT is implemented in JavaScript using the Heap  
ADT provided in the attached PriorityQueue.js and Heap.js. Note that the Heap  
stores keys (only elements), but the PQ stores items, i.e., (key, element)  
items/pairs so Item.js is also needed. Your task is to implement in JavaScript  
PQ-Sort based on the Priority Queue provided. Test it as before using the  
ArraySort-tests.js file, but note that it uses the HW07-ArraySorter.js from the  
previous assignment. You will also need to include the following statement at the  
top of your HW07-ArraySorter.js file:   
The ArraySort-tests.js file expects an implementation of your PQSort(arr)  
function so it can be called (see the provided file).**

**E. What can you conclude about the different sort algorithms?**

**C-4.13 Suppose we are given two sequences A and B of *n* elements, possibly containing  
duplicates, on which a total order relation is defined (i.e., has a comparator). Using a  
Priority Queue design an efficient pseudo-code algorithm for determining if A and B  
contain the same set of elements (possibly in different orders and possibly containing  
duplicates). What is the running time of this method?  
Implement your solution to C-4.13 in JavaScript and create some tests.**

Algorithm isSimilar(A, B)

if A.size() != B.size() then

return false;

PQ1 := new Priority Queue

PQ2 := new Priority Queue

while ! A.isEmpty() do

e1 := A.remove(A.first())

e2:= B.remove(B.first())

PQ1.insertItem(e1,e1)

PQ2.insertItem(e2, e2)

while PQ1.size() > 0 do

e1 := P1.removeMin()

e2:= P2.removeMin()

A.insertLast(e1)

B.insertLast(e2)

return compareSet(A, B)

Algorithm compareSet(A, B)

a := A.first()

b:= B.first()

while ( ! (A.isLast(a) ) do

if a.element() != b.element() then

return false;

else

a =A.after(a)

b =B.after(b)

if a.element() != b.element() then return false

return true;