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.
B. Is the bucket-sort algorithm in-place? Why or why not?
C. Illustrate the performance of the radix-sort algorithm on the following input sequence (22, 15, 26, 44, 10, 3, 9, 13, 29, 25).
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:

```
const PQ = require('./PriorityQueue.js');
```

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.