Assignment 3

Grades: each assignment is 5% in the final score and there is a 2% bonus in each assignment for bonus problems. We use the following scale for the simplicity of grading: the total score is 50 (additional 20 for bonus problems) for each assignment.

Problem 1 [Full Score: 10]. Show the contents of an array A implementing a binary min-heap. Repeatedly insert the following values into an initially empty heap: 7, 6, 5, 4, 3, 2, 1. Use the up-heap bubbling insertion of the items one-by-one. (Show the content after each insertion.)

Problem 2 [Full Score: 10]. Show the contents of the array A as a binary min-heap, initially with entries [7, 6, 5, 4, 3, 2, 1], as it changes during the linear time bottom-up heap construction. (Show the content after each down-heap bubbling.)

Problem 3 [Full Score: 10]. C-2.32 (on page 135). (You need to describe your algorithm and show why its complexity is $O(k)$.)

Problem 4 [Full Score: 10]. Given $k$ sorted queues (i.e., the front of each queue stores the minimum element in each queue. ) containing a total of $n$ elements, describe how to merge them into a single sorted queue in time $O(n \log k)$. You need to describe your algorithm and prove its complexity. (Hint: use a heap.)

Problem 5 [Full Score: 10]. Suppose we changed the height-balance property, so that the heights of two subtrees may differ by at most 2 (instead of 1). Prove that this modified AVL tree must still have a height that is $O(\log n)$. Please make your proof clear and formal. HINT: I recommend using the proof of Theorem 3.2 (page 153) as a starting point.

Problem 6 [Bonus Problem: 20].
(1) [10] C-2.31 (on page 135). (You need to describe your algorithm and show why its complexity is $O(n + k \log(n))$.)
(2) [10] Describe a data structure that supports both removeMin() and removeMax() with $O(\log(n))$ complexity. You need to show the implementation of insert(), removeMin(), and removeMax(), and prove their complexity. Hint: one can build on top of the heap.