1 Give an $O(n \log k)$ time algorithm to merge $k$ sorted lists into one sorted list, where $n$ is the total number of elements in all the input lists. (Hint: Use a min-heap for $k$-way merging.) (15 points)

2 What is the worst-case running time for the bucket-sort algorithm? What simple change to the algorithm preserves its linear expected running time and makes its worst-case running time $O(n \log n)$? (15 points)

3 What does it mean for a sorting algorithm to be stable? Prove that COUNTING-SORT is stable. Which of the following sorting algorithms are stable: insertion sort, merge sort and heapsort? Give a simple scheme that makes any sorting algorithm stable. How much additional time and space does your scheme entail? (20 points)

4 Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined? Explain by drawing tree for each case. (15 points)
   b. 924, 220, 911, 244, 898, 258, 362, 363.

5 Consider a node in a binary search tree that has two children. What can you say about its successor’s left child and its predecessor’s right child. Explain your your claims. (10 points)

6 Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key $k$ in a binary search tree ends up in a leaf. Consider three sets: $A$, the keys to the left of the search path; $B$, the keys on the search path; and $C$, the keys to the right of the search path. Professor Bunyan claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Prove this or give a smallest possible counterexample to the professor’s claim. (10 points)

7 Is the operation of deletion “commutative” in the sense that deleting $x$ and then $y$ from a binary search tree leaves the same tree as deleting $y$ and then $x$? Argue why it is or give a counterexample. (15 points)