Problem 1. Prove: A tree with $n$ nodes has $n - 1$ edges.

Problem 2 (Max Heap). The binary heap we discussed in the class is called the min-heap because of the delete-min operation. Conversely, a max-heap on a set $S$ of integers aims to support insertions and the following delete-max operation:

- **Delete-max**: Reports the largest integer in $S$, and removes it from $S$.

Describe how a min-heap can be used to implement a max-heap without changing its structure and algorithms. Your max-heap must still use $O(|S|)$ space, and support an insertion and a delete-max operation in $O(\log |S|)$ time.

Problem 3. Let $A$ be an array of length $n$ that stores a set $S$ of $n$ integers. The array is not sorted. Give an algorithm to find the $\sqrt{n}$-th smallest integer in $S$. Your algorithm must terminate in $O(n)$ worst-case time.

Problem 4* (Priority Queue with Attrition). Let $S$ be a dynamic set of integers. At the beginning $S$ is empty. We want to support the following operations:

- **Insert-with-Attrition**($e$): First removes all integers in $S$ that are greater than $e$, and then adds $e$ to $S$.
- **Delete-Min**: Removes and returns the smallest integer of $S$.

For example, suppose we perform the following sequence of operations:

1. Insert-with-Attrition(83)
2. Insert-with-Attrition(5)
3. Insert-with-Attrition(10)
4. Insert-with-Attrition(15)
5. Insert-with-Attrition(12)
6. Delete-Min
7. Delete-Min

After Operation 3, $S = \{5, 10\}$ (note that 83 has been deleted by Operation 2). After Operation 5, $S = \{5, 10, 12\}$. After Operation 6, $S = \{10, 12\}$.

Describe a data structure with the following guarantees:

- At all times, the space consumption is $O(|S|)$.
- Any sequence of $n$ operations (each being an insert-with-attrition or delete-min) is processed with $O(n)$ time, i.e., $O(1)$ amortized time per operation.

Problem 5 (Textbook Exercise 6.5-9). Suppose that we have $k$ arrays $A_1, A_2, ..., A_k$ of integers, such that each array has been sorted in ascending order. Let $n$ be the total number of
integers in those arrays. Describe an algorithm to produce an array that sorts all the $n$ integers in ascending order (you may assume that no integer exists in two arrays). Your algorithm must finish in $O(n \log k)$ time.

For example, suppose that $k = 3$, and that the three arrays are $(2, 23, 32, 35, 37), (5, 10), \text{ and } (33, 58, 82)$. Then you should produce an array containing $(2, 5, 10, 23, 32, 33, 35, 37, 58, 82)$. 