Problem 1 (30 points)

Describe and analyze a data structure that supports the following operations.

- INSERT($x$): Given an element $x$, insert it into the data structure. This should take constant time.
- DELETE($x$): Delete $x$ from the tree. This should take constant time.
- RETURN($x$): Return an element $x$ such that its rank satisfies:

\[
\frac{1}{2} n - \frac{1}{100} n \leq \text{rank}(x) \leq \frac{1}{2} n + \frac{1}{100} n
\]

where $n$ is the number of elements in the data structure (at the time RETURN is called).

Prove that the amortized time for any sequence of operations is constant.

Problem 2 (15 points)

Construct a fast algorithm that finds the largest, second largest and third largest elements in an array $A$. Analyze the running time of your algorithm.

*Hint: Remember what we did in class.*

Problem 3 (10 points)

Prove that the running time of heapsort on an array $A$ that is sorted in ascending order is $\Omega(n \log n)$. Prove the same bound when the array is in decreasing order.

Problem 4 (20 points)

Prove that no matter what node we start at in a height $h$ binary search tree, $k$ successive calls to TREE-SUCCESSOR take $O(h + k)$ time.

Problem 5 (10 points)

If $x$ is a non-root node in a binomial tree within a binomial heap, and the degree of parent($x$) is $l$, then what values can the degree of $x$ take and what does this value depend on.