Problem 1. Construct a comparison based fast algorithm that finds the largest, second largest and third largest elements in an array A. Analyze the running time of your algorithm.

(15 points) if running time is at most \( n + 5 \log n \).
(20 points) if you can achieve running time at most \( n + 3 \log n \).
(25 points) if you achieve at most \( n + \frac{5}{2} \log n \) running time.

Problem 2. (15 points) Suppose that you are given a \( k \)-sorted array, in which no element is farther than \( k \) positions away from its final (sorted) position. Show that worst case running time of any comparison based algorithm for sorting \( k \)-sorted array is \( \Omega( n \log k ) \) comparisons.

Problem 3. (30 points) Consider a data structure called LazyHeap that supports the following operations:

- \text{INSERT}(x)\): Given an element \( x \), insert it into the data structure. It has no cost.
- \text{DELETE}(x)\): Delete \( x \) from the data structure. It has no cost.
- \text{RETURN}(x)\): Return an element \( x \) such that its order, if the elements are sorted, satisfies:

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

,where \( k \) is the number of elements in the data structure (at the time \text{RETURN} is called). \text{RETURN} also has no cost.

(Sometimes you have to make comparisons between 2 elements so that you can perform \text{RETURN} operation correctly. 1 comparison costs 1 unit.) Come up with a strategy for the comparisons so that the running time for any sequence of \( n \) operations is less than \( 1000n \).

Problem 4. Consider an array of \( (k + 1) \) digits, in which we store binary digits (each digit can contain 0 or 1). Assume that the right most digit is the least significant one. Let \( n = 2^\ell \), for some integer \( \ell \) such that \( \ell < k \).

(10 points) Give an upper bound for a total cost if we start from 0, if first \( n \) operations are \text{INCREMENT}, and last \( n \) operations are \text{DECREMENT}(changing a bit costs 1 unit).

(10 points) Construct a sequence of \( 2n \) \text{INCREMENT} and \text{DECREMENT} operations starting from 0 such that the total cost is \( \Omega(\ell n) \).