Please start each problem on a new page, and include your name on each problem.

Remember: you may work in groups of up to three people, but must write up your solution entirely on your own. Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. You may use the internet to look up formulas, definitions, etc., but may not simply look up the answers online.

Please include proofs with all of your answers, unless stated otherwise.

1 Jug Matching (33 points)

Suppose that you are given \( n \) red and \( n \) blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do all the blue ones. Moreover, for every red jug there is a blue jug that holds exactly the same amount of water (and vice versa).

Your job is to find a matching between red jugs and blue jugs that hold the same amount of water. To do this, you are only allowed to use the following operation: pick a red jug and a blue jug, fill the red jug, and pour it into the blue jug. This will tell you whether the volume of the red jug is less than, equal to, or greater than the volume of the blue jug. In other words, you cannot compare two red jugs or two blue jugs.

(a) Prove that any algorithm for this problem must make \( \Omega(n \log n) \) comparisons in the worst case.

(b) Give a randomized algorithm that uses \( O(n \log n) \) comparisons in expectation, and prove that this bound is correct.

2 Median of Sorted Arrays (33 points)

Let \( A \) and \( B \) be sorted arrays of \( n \) elements each. We can easily find the median of \( A \) or the median of \( B \), since they are already sorted – it will be at index \( \lfloor \frac{n+1}{2} \rfloor \). But what if we want to find the median element of \( A \cup B \)? We could just concatenate them and use the \( O(n) \) time median algorithm, but is it possible to do better? In this problem you should give matching upper and lower bounds, i.e. you should find a function \( f(n) \) and do the following:

(a) Design a deterministic algorithm whose running time (measured in terms of the number of comparisons) is \( O(f(n)) \), and

(b) Give a lower bound showing that any comparison-based algorithm must make \( \Omega(f(n)) \) comparisons in the worst case.
3 Non-comparison sorting (34 pts)

(a) Suppose we are given an array of integers, but instead of all integers having the same length they can each have a different number of bits. So e.g. the number 0 or 1 takes one bit, the numbers 2, 3 take 2 bits, the numbers 4, 5, 6, 7 take three bits, etc. However, the total number of bits over all of the integers in the array is equal to $n$. Show how to sort the array in $O(n)$ time.

(b) Suppose now that we are given an array of strings (over some finite alphabet, say the letters a-z). Each string can have a different number of characters, but the total number of characters in all the strings (i.e. the sum over the strings of the length of the string) is equal to $n$. Show to sort the strings lexicographically in $O(n)$ time. Here lexicographic order is the standard alphabetic order, so for example $a < ab < b$. 