COL106: Data Structures. I semester, 2021-22. Major Exam.
18 November 2021. Maximum Marks: 40 marks

Name	I	**Ent. No.**

Exam Instructions for offline students.
1. There are 3 sheets of paper. Write your name and entry number in the given space on each of the two sheets.
2. Answer clearly *only in the space provided.* Do NOT write on this page. **No extra sheet will be given or collected**.
2. Workout your answers in rough and then transfer carefully to the printed sheet.
3. Rough sheets will **not** be collected.
4. If you need a new printed sheet, you can get one but you will have to transfer your entire exam onto the new sheet. We will collect only three sheets of paper from you.

Problem 1 (8 marks) Explain in words (**no pseudocode**) how to delete *any* element from the heap (not just the min). You can assume that (a) all elements in the heap are distinct, (b) the element to be deleted is actually present in the heap and (c) that you are given a pointer to the location of the element that you have to delete. You *must* argue for the correctness of your algorithm and tell us, with justification, its running time.

Problem 2 (4+4 marks) In an AVL tree for all nodes the difference between the heights of the left subtree and right subtree of x is at most 1. Suppose we relax this and allow the difference between the heights of the subtrees to be at most 2. We will call such BSTs *relaxed AVL trees.*

Problem 2.1 Let $\chi(h)$ denote the minimum number of nodes in a relaxed AVL tree of height h. Give a recurrence relation for computing $\chi(h)$, and compute $\chi(6)$ (include all intermediate calculations). Note: You do *not* have to solve the recurrence. You just have to write the recurrence *with justification* and then use the recurrence to compute $\chi(6)$.

Problem 2.2 Let $\tau(h)$ denote the minimum depth of a leaf node in a relaxed AVL tree of height h. Give a recurrence relation for computing $\tau(h)$. Note that you have to justify why this recurrence is correct. Solve the recurrence to find the exact value of $\tau(h)$ for general h.

Problem 3 (5+3 marks) In class, we saw that any comparison-based sorting algorithm must perform $\Omega(n \log n)$ comparisons. In this problem, we will consider the problem of *almost-sorting* an array. Given an array $A = [a_1, a_2, \ldots, a_n]$, we say that A is almost-sorted if for all $i < n - \sqrt{n}$, $a_i \leq a_j$ for all $j > i$. That is, the first $n - \sqrt{n}$ elements are in sorted order, and the last $\sqrt{n}$ elements are at least as large as the first $n - \sqrt{n}$ elements. Example: [3, 5, 6, 7, 9, 11, 12, 14, 15, 17, 18, 20, 41, 33, 25, 38] is an almost-sorted array.

Problem 3.1 Prove that there does not exist a comparison based algorithm that takes as input an array A, performs $O(n)$ comparisons, and outputs an almost-sorted array. Note that you are expected to write a formal proof.

Problem 3.2 Suppose the input array consists of **positive integers**, and you are given that at least $n - \sqrt{n}$ entries are in the range $[1, 100]$. Give an $O(n)$ time algorithm that outputs an almost-sorted array. You must argue for the correctness of your algorithm and its running time.

Problem 4 (8 marks) Let T be a 2-4 tree with n keys and height $h = O(\log n)$. Explain in words (**no pseudocode**) how to find the k^{th} largest key in T in $O(h)$ operations. You can assume that all keys are distinct. You may need to store additional attributes at every node in order to solve this problem. Explain clearly how you will augment the 2-4 tree and then explain your algorithm.

Problem 5 (8 marks) Given an unweighted, undirected graph $G = (V, E)$ in adjacency-list representation, construct a data structure $\mathcal{D}_G$ of size $O(n)$ such that the following queries can be answered in $O(1)$ time:

- Query(i, j) : outputs 1 if there exists a path from vertex i to vertex j in G, else outputs 0

You can assume that each vertex has a unique label in $\{1, 2, \ldots, n\}$. Explain in words (**no pseudocode**) (a) how to construct the above data structure, (b) the time required to construct the data structure $\mathcal{D}_G$, (c) the time required to answer queries.