Indian Institute of Technology Kharagpur
Department of Computer Science and Engineering

Algorithms-I (CS21003) **Class Test – 1 [Maximum Marks: 20]** **Spring Semester, 2019-2020**

Date: 03-Feb-2020 (Monday) | Time: 7:00pm – 8:00pm | Venue: F-116 / F-142

Name: _____ **Roll No:** _____

[Instructions: ***Write your answers in proper places mentioned in the question paper itself. Answer ALL questions. Be brief and precise. If you use any algorithm/result/formula covered in class, just mention it, do not elaborate.*** **]**

Q1. Let two recursive algorithms satisfy the following two recurrence relations:

(i) $T(n) = \begin{cases} 3.T(\frac{n}{3}) + n, & \text{if } n > 1 \\ 1 \quad , & \text{if } n = 1 \end{cases}$ and (ii) $T(n) = \begin{cases} \sqrt{n}.T(\sqrt{n}) + n.\log_2^d n, & \text{if } n > 2 \\ 2 \quad , & \text{if } n = 2 \end{cases}$ $(d \geq 0)$

Deduce the running time $T(n)$ in asymptotic Θ-notation for both of these cases separately. **[Marks: 4 + 6 = 10]**

Solution – (i):

ROUGH WORK

Solution – (ii):

ROUGH WORK

Q2. Let $\mathcal{A}$ be an $n \times n$ two-dimensional array with all distinct elements, in which all rows and all columns are sorted in ascending order from smaller to larger indices. Given a key x, your task is to find out whether x is present in $\mathcal{A}$.

(i) Propose a recursive formulation to solve this, from which you can design a $\Theta(n \log_2 n)$-time algorithm.
(ii) Propose an *efficient* recursive formulation to solve this, from which you can design a $O(n)$-time algorithm.

In both the above cases (separately), develop the recurrence relations from your recursive formulations and finally solve these to deduce the above-mentioned time-complexity of the algorithms. **[Marks: 4 + 6 = 10]**

Solution – (i):

ROUGH WORK

Solution – (ii):

ROUGH WORK