Department of Computer Science **UET Lahore, New Campus**			**Registration Nos:** ____________ ____________ ____________
Complex Computing Problem	**CSC-208 Design and Analysis of Algorithms**	**Deadline:** **6th April 2025**	**Total Marks: 20** **Marks Obtained:**

Note:

- You can work in a group of 3 or less.
- One assignment per group is required. Submit the code and a small document comprising the flow of your system on google classroom.
- The deadline for this problem is 6th April 2024.
- Every member of the group must be present at the time of evaluation.
- Vivas will be conducted from 6th April 2025.

==

Complex Computing Problem [Recurrence Solver]

No.	Attributes	Description
WP1	**Depth of Knowledge Required**	This problem requires a deep understanding of recurrence relations and recursive algorithms. The background knowledge of time and space complexity and familiarity of asymptotic notations and bound (lower, upper, and tight) is a must.
WP2	**Range of Conflicting Requirements**	This problem involves differing recurrence equations with decreasing or dividing functions or subproblems of different sizes.
WP7	**Interdependence**	The recurrence solver has dependance on sub solvers determined by the type of recurrence.

[Mapping CLOs: CLO2: 5 CLO3: 5 CLO4: 10]

Evaluation

Assessment Rubrics:

	Implementation				**Evaluation**	**Analysis**	
Roll No.	**Prompt**	**Identifying the type of recurrence**	**Selection of proper solver**	**Implementation of cases for each solver**	**Correct output**	**Identification of efficiency class**	**Run time with appropriate asymptotic notation**
	1	2	2	5	5	2	3

Recurrences	**Method**	**Results**	**System Solver**	**System O/P**

Problem Description: Recursive algorithms run time is computed using various methods like mathematical induction, substitution method, recurrence tree method or Master theorem etc. Each method has some limitations, as in substitution method some approximations are made and results are computed in Big-O notation, recurrence tree generates the guess which is then confirmed using substitution method or any other method. Master theorem mentioned in Book *Introduction to Algorithms (CLRS MIT Press, 2024)* guarantees a tight bound to a recurrence equation if some constraints are satisfied. Also, the recurrence equation for different subproblem sizes is either calculated on shallowest branch (lower bound) or the deepest branch (upper bound).
Using your theoretical knowledge of recurrences and methods to solve recurrences, you are required to design and develop a **Recurrence Solver** capable of solving recursive equations formulated from recursive algorithms' basic operation. Recursive equations can have different subproblem sizes or can have sub problems of same size. Recurrences can have a dividing function or a decreasing function. Your solver must be capable of handling any type of recurrence equations and give the running time with an appropriate asymptotic notation.

- General form of recurrence equations with dividing function:
  - a sub problems of $\frac{n}{b}$ size: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$
  - Sub problems of different sizes: $T(n) = T\left(\frac{n}{b}\right) + T\left(\frac{n}{b'}\right) + f(n)$
- General form of recurrence equations with decreasing function:
  - $T(n) = a\,T(n-b) + f(n)$

Requirement: Your system must prompt the user for cases i.e. dividing function or decreasing function. Next it must prompt for the values of $a, b \text{ and } f(n)$. It must compare the values and select amongst the methods for recurrence solver i.e. Master theorem, Extended Master theorem, Muster theorem and "your approximation method for a recurrence having sub problems of different sizes".

Expected output: The run time of the recursive algorithm is *efficiency class* and $T(n)$ = O (…) or $T(n) =$ Θ (…) or $T(n) = \Omega$ (…).