

**ASSIGNMENT 01**

**Design Analysis and Algorithms**

**SUBMITTED BY:**

* Muhammad Saad Cheema (2023-CS-729)
* Muhammad Shakeel (2023-CS-736)

**SUBMITTED TO**

Ma'am Drakshan Bokht

**DEPARTMENT OF COMPUTER SCIENCE**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY LAHORE, NEW CAMPUS, KSK**

**Project Title:**

**Smart Recurrence Solver**

# **Objective**

To build an interactive Python-based system that automatically detects and solves recurrence relations using techniques like the Master Theorem, substitution, and approximation, while supporting multiple notation preferences (Θ, O, Ω).

# **System Flow**

1. Start Program  
- Display a stylized welcome screen.  
  
2. Notation Selection  
- Prompt user to select preferred asymptotic notation:  
 - Θ (Big-Theta)  
 - O (Big-O)  
 - Ω (Big-Omega)  
  
3. Input Phase  
- User enters recurrence relation in the format: Example: T(n) = 2\*T(n/2) + n  
  
4. Recurrence Detection  
- The system parses the input and identifies the type:  
 - Dividing (e.g., T(n) = a\*T(n/b) + f(n))  
 - Decreasing (e.g., T(n) = T(n-1) + f(n))  
 - Mixed (e.g., T(n) = T(n/3) + T(2n/3) + f(n))  
  
5. Solving Logic  
- Dividing: Apply Master Theorem (Case 1, 2, 3).  
- Decreasing: Use iterative/substitution logic.  
- Mixed Sizes: Approximate with logarithmic-based estimation.  
  
6. Output  
- Final result printed with the selected notation, e.g., T(n) = Θ(n log n)  
  
7. Repeat or quit  
- The user can solve multiple recurrences or type "quit" to exit.

# **Technologies Used**

- Python 3  
- SymPy (symbolic computation)  
- Math module