Generic Lab (Experiment 3):

Program-1

Write a program in C++/ C program that implements the Master Theorem using a switch case structure. It allows the user to input the values for a, b, c, and k, and then selects the appropriate case (1, 2, or 3) to determine the asymptotic behavior of the recurrence relation. The program also includes a case for when the Master Theorem does not apply.

Hint:

logBase(a, b) Function:

- **Purpose:** Calculates the logarithm of a with base b.

f_n(n, c, k, type) Function:

- **Purpose:** Simulates the behavior of the function f(n) for different cases of the Master Theorem.
- **Cases:**
    - **Case 1:** Simulates f(n) as O(n^log_b(a) - ε).
    - **Case 2:** Simulates f(n) as Θ(n^log_b(a) * log^k(n)).
    - **Case 3:** Simulates f(n) as Ω(n^log_b(a) + ε).
    - **Case 4:** The regularity condition failing or `f(n)` not fitting any of the specific forms mentioned in the theorem. Master's Theorem can't be applied to every recurrence relation of the form T(n) = aT(n/b) + f(n) It can't be applied **when a is less than 1, and a is not a constant**, i.e. a= x. It can't be applied when f(n) is not a polynomial and T(n) is not monotonic.

main Function:

- **Input:**
    - Prompts the user to input values for a, b, c, k, and caseType.
- **Calculation:**
    - Compute log_b_a using the logBase function.
- **Decision:**
    - Uses a SWITCH statement to determine and print the asymptotic behavior based on the selected case (caseType).
- **Output:**
    - Prints the appropriate asymptotic behavior for the selected case.

*****Solve the below problem by using pen and paper (Take a screenshot and submit it with your code)**

T (n) = 8 T * $\left(\frac{n}{2}\right) + 1000n^2$ **apply the master theorem on it.**

<u>Program-2</u>

Write a program in C++/ C that constructs and prints a recursion tree based on user inputs.

<u>Hint:</u>

Definitions and Structures:

- MAX_NODES defines the maximum number of nodes that the queue can handle.
- The Node structure holds size (the problem size) and level (the depth level).

Input:

- The user inputs the values for a (number of subproblems), b (division factor for problem size), and depth (maximum depth of the recursion tree).

Initialization:

- The problem size n is initialized to 100.
- The queue array and indices front and rear manage nodes.

Building and Printing the Tree:

- Enqueue the initial node with size n and depth 0.
- While there are nodes in the queue:
    - Dequeue a node and print its level and size with indentation based on the node's depth.
    - If the current depth is less than the maximum depth:
        - Enqueue child nodes with updated problem size (size / b) and incremented depth.

*****Solve the below problem by using pen and paper (Take a screenshot and submit it with your code)**

$T(n) = T(n/4) + T(n/2) + n^2$ apply the recursion tree method.