**Artificial Intelligence (AI) Lab Report**

**Name:** Aqsa Gillani  
**Roll No:** 064  
**Course:** Artificial Intelligence (AI) Lab  
**Semester:** Fall 2025

**1. DFS with Stack**

**Objective:**  
To implement Depth First Search (DFS) using an explicit stack data structure.

**Description:**  
This program explores a graph using DFS implemented iteratively with a stack instead of recursion. It visits nodes deeply before backtracking.

**Code Explanation:**

* A stack is used instead of recursion to keep track of nodes.
* The algorithm starts from the given node, pushes it onto the stack, and processes it.
* Each unvisited neighbor is pushed into the stack (in reverse order to maintain correct traversal).
* Nodes are marked as visited when popped from the stack.

**Output:**



**2. Inorder, Preorder, Postorder Traversals in DFS**

**Objective:**  
To implement DFS traversals (Inorder, Preorder, Postorder) on a binary tree.

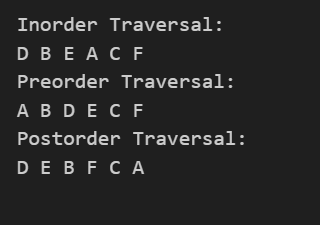
**Description:**  
These traversals are DFS variations applied to binary trees:

* **Inorder:** Visits Left → Root → Right
* **Preorder:** Visits Root → Left → Right
* **Postorder:** Visits Left → Right → Root

**Code Explanation:**

* A binary tree Node class is defined with key, left, and right attributes.
* Recursive functions are written for Inorder, Preorder, and Postorder traversals.
* Each function prints nodes in its specific order.

**Output:**



**3. BFS without Queue & without Node**

**Objective:**  
To implement Breadth First Search (BFS) without using a Queue or Node class.

**Description:**  
BFS is implemented using two lists (current\_level and next\_level) to simulate queue behavior. The graph is represented as an adjacency list.

**Code Explanation:**

* The algorithm starts at the root node and processes all nodes at the current level.
* Each level is processed using a list (current\_level).
* For each visited node, unvisited neighbors are added into next\_level.
* The loop continues until all levels are processed.

**Output:**  


**4. BFS with Queue**

**Objective:**  
To implement BFS using a queue data structure.

**Description:**  
BFS is implemented with a queue to process nodes in **FIFO order**, ensuring level-order traversal of a graph represented as adjacency list.

**Code Explanation:**

* A queue is initialized with the start node.
* The algorithm removes nodes from the front of the queue (FIFO).
* Each visited node is processed and its unvisited neighbors are added to the queue.
* This ensures that nodes are visited level by level.

**Output:**  
