**LAB 5:**

The code implements the **Depth-First Search (DFS) algorithm** using an **iterative approach** with a **stack** to traverse a graph. DFS is a common graph traversal technique that explores as far as possible along a branch before backtracking.

**Explanation of the Code**

1. **Initialize the Stack and Visited List**
   * The function dfs(graph, start\_node) takes a graph (dictionary) and a starting node as input.
   * It initializes a stack with the start\_node and an empty list visited to track explored nodes.
2. **Iterative DFS Execution**
   * The algorithm enters a **while loop**, which runs until the stack is empty.
   * The **last added node** (current\_node) is popped from the stack (**LIFO order**).
   * If it has not been visited, it is added to the visited list and printed.
   * The adjacent nodes of current\_node (from graph[current\_node]) are **reversed** and added to the stack to maintain **correct DFS order**.
3. **Graph Representation & Execution**
   * The graph is represented as an **adjacency list** (a dictionary).
   * Running dfs(new\_graph, 'a') starts traversal from 'a'.
   * The expected DFS traversal order for this graph is:
   * a b d e f c
   * Note that DFS order may vary based on how neighbors are stored.

**Key Takeaways**

**Uses an explicit stack** instead of recursion.  
 **Traverses graphs in depth-first order (LIFO mechanism).**  
 **Handles both directed and undirected graphs.**

**Efficient for pathfinding in tree/graph structures.**