**DFS – Graph and Tree Traversals Lab**

Name: AYESHA SIKANDAR  
Roll No.: 044  
**Section: BSDS 3A**

**1. DFS – Graph Traversal using Stack**

**Code Explanation:**

I created a function dfs\_stack() to perform Depth-First Search on a graph using a stack.  
The graph is represented as a dictionary, with each key as a node and its value as a list of neighbors.  
I use a list called stack to keep track of nodes to visit and a list visited to store visited nodes.  
Inside a while loop, I pop the last node from the stack. If the node is not visited, I add it to visited and print it.  
I then add its neighbors to the stack in reversed order so the traversal order matches recursive DFS.  
The loop continues until all nodes are visited.

**How I Made It:**

I used basic Python features like lists, loops, and conditions.  
I used a stack to explore nodes in a depth-first order.  
Reversing the neighbors ensures that nodes are visited in the same order as recursive DFS.

**Why I Made It:**

I made this to understand DFS traversal in graphs.  
It helps me learn how nodes are explored deep first and how stacks are used in iterative graph algorithms.

**2. Tree DFS Traversals – Preorder, Inorder, Postorder**

**Code Explanation:**

I represented a tree using a dictionary, where each node has keys 'left' and 'right' for its children.  
I wrote three recursive functions:

* preorder() → visits root first, then left, then right.
* inorder() → visits left first, then root, then right.
* postorder() → visits left and right first, then root.  
  Each function prints nodes when visited to show the traversal order.
* How I Made It:
* I used dictionaries and recursion to traverse the tree.  
  Each traversal has a separate function to clearly demonstrate the different DFS orders.  
  I avoided Node classes because the task required implementation without nodes.
* Why I Made It:
* I made this to understand DFS on trees in different orders.  
  Preorder helps with tree copying, Inorder is useful for sorting (BST), and Postorder is helpful for deletion operations.  
  It also helped me practice recursion in Python.

**How I Made It:**

I used **dictionaries and recursion** to traverse the tree.  
Each traversal has a separate function to clearly demonstrate the **different DFS orders**.  
I avoided Node classes because the task required implementation **without nodes**.

**Why I Made It:**

I made this to understand **DFS on trees** in different orders.  
Preorder helps with tree copying, Inorder is useful for sorting (BST), and Postorder is helpful for deletion operations.  
It also helped me practice recursion in Python.