

**Name:**  Muhammad Awais Asghar

**Roll Number:** SU92-BSAIM-F24-065

**SUBJECT:** Artifical Intelligence (LAB)

**Task No -5**

**Question 1:**

**Why this code was made:**

This code's aim is to use a stack rather than a recursive method to implement Depth First Search (DFS) on a graph.  
A basic graph analysis method called DFS is used to methodically examine every node and edge in a graph.  
This method works well in practical applications such as:  
  
Finding your way within difficulties or maps  
  
Finding network cycles  
  
Resolving puzzles  
  
analyzing connections within data structures or social networks

**How this code works:**

Graph Illustration

* Each key in the dictionary represents a point in the graph, and its value is a list of connected points.

Set Visited

* A set called visited keeps track of all elements that have already been explored, preventing revisits.

Initialization of the Stack

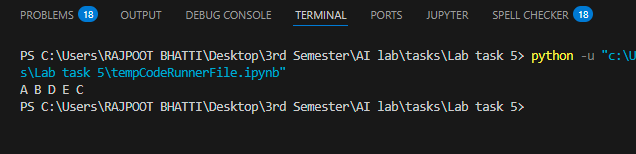
* The stack is initialized with the starting element 'A'.
* The stack determines the order in which elements are explored.

Loop Traversal

* While the stack is not empty:
  1. Remove the last element from the stack (element).
  2. If this element has not been visited yet:
     + Print the element.
     + Add it to visited.
     + Push all of its neighbors onto the stack in reversed order, so that the leftmost neighbor is processed first, preserving DFS order.

Order of Traversal

* For the given graph\_data, starting from 'A', the DFS order printed



**Question 2:**

**Why this code was made:**

This code's goal is to use a binary tree in the preorder, inorder, and postorder common orders.  
A major concept in computer science, tree traversal is helpful in:  
  
Analyzing or searching through data structures  
  
Evaluation of compilers and expressions  
  
Organizational structures and file systems  
  
Many algorithmic issues, such as tree-based searches

**How this code works:**

Representation of Trees  
  
The tree can be saved as a dictionary, with 'left' and 'right' keys pointing to children and each key representing a node label.  
  
means that A has a left child, B, and a right child, C.  
  
Functions of Traversal  
  
Traversal in preorder (Start - Left - Right)  
  
Start with 'A'.  
  
The current element should be printed.  
  
Visit the left child continually.  
  
Visit the correct child recursively.  
  
Traversal in order (Left - Start - Right)  
  
Start with 'A'.  
  
Visit the left child recursively.  
  
The current element should be printed.  
  
Visit the appropriate child recursively.  
  
(Left - Right - Start) Postorder Traversal  
  
Start with 'A'.  
  
Visit the left child recursively.  
  
Visit the appropriate child recursively.  
  
The current element should be printed.  
  
Example of Traversal Order:  
  
The result using your my\_tree will be

