|  |  |
| --- | --- |
| Project Title : | Lab Task 5 |
| Name : | **Aman Ali** |
| Roll No : | **195** |
| Class : | BSAI |
| Section : | **3C** |
| Submission Date : | September 25, 2025 |

DFS With Stack

**Code Explanation**

* This code implements **Depth-First Search (DFS)** on a tree represented as a Python dictionary.
* The tree nodes are stored as keys, and their children are stored as lists of values.
* The dfs function uses a **stack** (stack\_new) to keep track of nodes to visit next.
* The visited list keeps track of nodes that have already been visited to avoid repetition.
* For each node, the function prints the node and adds its children to the stack in **reversed order** to maintain DFS traversal order.

**How It Works**

* Start from the **starting node** (s) and push it onto the stack.
* Loop until the stack is empty:
  + Pop the top node from the stack.
  + If the goal has not been visited yet:
    - Print the node and mark it as visited.
    - Add its children to the stack in reversed order.
* The DFS continues until all reachable nodes from the starting node are explored or the goal node is found.

**Why This Approach**

* Using a **stack** allows the algorithm to follow the **LIFO (Last In First Out)** order, which is key for DFS.
* Reversing the children before pushing ensures the nodes are visited in the intended order.
* Maintaining a **visited list** prevents cycles or repeated processing in larger graphs (important for general graphs).
* This approach is simple and efficient for small to medium-sized tree or graph structures.

**Sample Output**

