**AI LAB TASK 5**

**Explanation of DFS using STACK**

1. **visited = []**  
   Stores all nodes that have already been visited.
2. **stack = [start]**  
   The stack starts with the given start node. Stack works on **LIFO** (Last In First Out).
3. **while stack:**  
   Loop runs until the stack becomes empty.
4. **node = stack.pop()**  
   Takes the top element from the stack to explore.
5. **if node not in visited:**  
   Prevents revisiting the same node.
6. **visited.append(node)**  
   Marks the node as visited.
7. **stack.extend(reversed(graph[node]))**  
   Adds the node’s neighbors into the stack **in reverse order**, so that the left-most neighbor is processed first (correct DFS order).
8. **return visited**  
   Returns the final DFS traversal sequence.

**Why reversed(graph[node])?**

* DFS uses a stack, so the **last pushed element is visited first**.
* If we push neighbors in normal order, the **right-most neighbor** will be visited first.
* By reversing, the **left-most neighbor** comes on top of the stack, so it gets visited first.

'

**Step-by-Step Stack Operations**

1. Start at **A** → Stack = ['A']
2. Pop A → visit A → push neighbors B, C (reversed) → Stack = ['C', 'B']
3. Pop B → visit B → push neighbors D, E (reversed) → Stack = ['C', 'E', 'D']
4. Pop D → visit D → no neighbors → Stack = ['C', 'E']
5. Pop E → visit E → no neighbors → Stack = ['C']
6. Pop C → visit C → push neighbor F → Stack = ['F']
7. Pop F → visit F → push neighbor G → Stack = ['G']
8. Pop G → visit G → no neighbors → Stack = []

Final DFS Traversal = **A, B, D, E, C, F, G**