**Name: Laiba Adnan**

**Class: Bsds 3A**

**Roll no: 004**

**Task 5**

**DFS with Stack & Node**

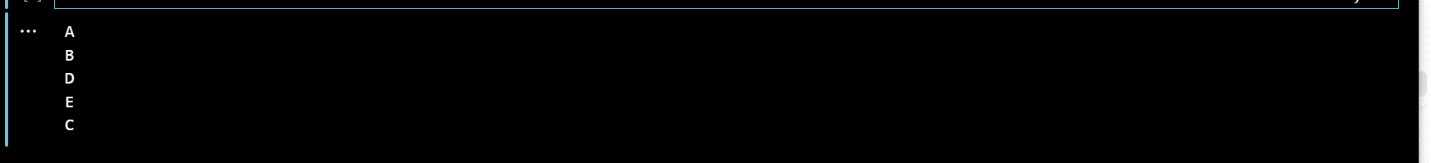
**⚙️ How It Works**

1. **Start**
   * stack = ['A'], visited = {}
2. **Pop A**
   * Print A
   * Add A to visited
   * Push neighbors of A in **reversed** order → first C, then B
   * Stack = ['C', 'B']
3. **Pop B**
   * Print B
   * Add B to visited
   * Push reversed neighbors of B → E, then D
   * Stack = ['C', 'E', 'D']
4. **Pop D**
   * Print D
   * No neighbors → Stack = ['C', 'E']
5. **Pop E**
   * Print E
   * No neighbors → Stack = ['C']
6. **Pop C**
   * Print C
   * No neighbors → Stack = []

**❓ Why reversed() is used**

We reverse the neighbor list so that **left-to-right DFS order** is maintained (because stack is LIFO).  
Without reversed(), it would explore C before B.

**Output:-**

****

**Q: Research about Inorder, Preorder, Postorder?** **Inorder, Preorder, and** Postorder are tree traversal methods based on Depth First Search (DFS). They define the order in which nodes are visited:

Preorder: Root → Left → Right

Inorder: Left → Root → Right

Postorder: Left → Right → Root

These methods are used to visit all nodes in a binary tree and are part of DFS traversal.

No implementation was required — only research.