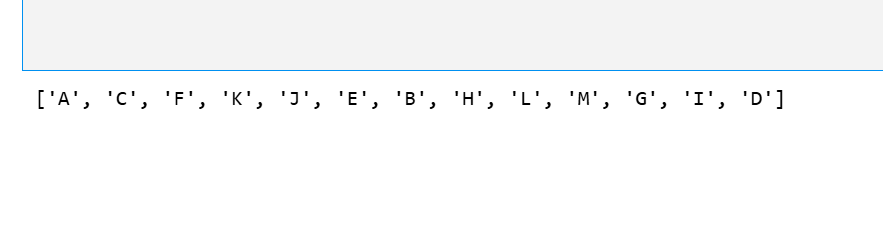
**Task # 5 Description**

This program performs a **Depth-First Search (DFS)** using a **stack (iterative method)** to explore all nodes in a graph

1. **Function dfs\_stack(graph, start)**
   * **visit** → a set to keep track of already visited nodes (avoiding repetition).
   * **stack** → initialized with the starting node (used instead of recursion).
   * **result** → stores the order in which nodes are visited.
2. **DFS Logic (Loop)**
   * While the stack isn’t empty:
     + Pop the **top node** (LIFO order).
     + If not visited →
       - Add it to result and mark it as visited.
       - Push all its connected nodes (**children**) into the stack using reversed() so that the first neighbor in the list gets visited first (maintains natural left-to-right order).
3. **Graph Example**  
   The given graph dictionary shows each node and its connections (edges).  
   For example:
   * 'A' connects to 'B' and 'C'
   * 'B' connects to 'D', 'G', 'H'
   * 'F' connects to 'J', 'K', etc.
4. **Output**
   * Starting from 'A', DFS explores as deep as possible before backtracking.
   * The printed list shows the **order of traversal** of all connected nodes

**Output Screen Shot:**

****