TASK 5

# DFS with Stack (Word Version):

This program demonstrates the Depth First Search (DFS) algorithm using an explicit stack. The code has been modified to use the word 'word' instead of 'node' for clarity.

## Step-by-Step Explanation:

1. A function named dfs\_stack is defined. It takes two arguments:

- graph: A dictionary representing the graph.

- start: The starting word for DFS.

2. Inside the function, a set named visited is created to keep track of the words that have already been visited.

3. A list named stack is initialized with the starting word. This stack will help us explore the graph step by step.

4. The algorithm runs a while loop as long as the stack is not empty:

- The last word is popped from the stack.

- If the word has not been visited yet, it is printed and added to the visited set.

- The neighbors of this word (taken from the graph dictionary) are reversed and checked. If a neighbor has not been visited yet, it is pushed onto the stack.

5. This process continues until all words connected to the starting word are visited.

## Graph Used in the Program:

The graph is represented as a dictionary where each key is a word and the value is a list of neighboring words:

{  
 'A': ['B', 'C'],  
 'B': ['D', 'E'],  
 'C': [],  
 'D': [],  
 'E': []  
}

Here, 'A' connects to 'B' and 'C'. 'B' connects to 'D' and 'E'. 'C', 'D', and 'E' have no further connections.

## Output of the Program:

When we call dfs\_stack(graph, 'A'), the program starts from 'A' and explores as deep as possible along each branch before backtracking. The printed result will be:

A B D E C