def dfs(graph, start\_node):  
    visited = set()  
    traversal\_order = []  
  
    def dfs\_helper(node):  
        visited.add(node)  
        traversal\_order.append(node)  
  
        for neighbor in graph[node]:  
            if neighbor not in visited:  
                dfs\_helper(neighbor)  
  
    dfs\_helper(start\_node)  
    return traversal\_order  
  
def bfs(graph, start\_node):  
    visited = set()  
    traversal\_order = []  
    queue = [start\_node]  
  
    while queue:  
        node = queue.pop(0)  
        if node not in visited:  
            visited.add(node)  
            traversal\_order.append(node)  
  
            for neighbor in graph[node]:  
                if neighbor not in visited:  
                    queue.append(neighbor)  
  
    return traversal\_order  
  
def main():  
    num\_nodes = int(input("Enter the number of nodes: "))  
    graph = {}  
  
    for i in range(num\_nodes):  
        node = input(f"Enter node {i+1}: ")  
        neighbors = input(f"Enter neighbors of node {node} (space-separated): ").split()  
        graph[node] = neighbors  
  
    start\_node = input("Enter the start node: ")  
  
    print("DFS Traversal Order:")  
    print(dfs(graph, start\_node))  
  
    print("\nBFS Traversal Order:")  
    print(bfs(graph, start\_node))  
  
if \_\_name\_\_ == "\_\_main\_\_":  
    main()

OUTPUT:

Enter the number of nodes: 5  
Enter node 1: 1  
Enter neighbors of node 1 (space-separated): 2 3  
Enter node 2: 2  
Enter neighbors of node 2 (space-separated): 4 5  
Enter node 3: 3  
Enter neighbors of node 3 (space-separated): 4  
Enter node 4: 4  
Enter neighbors of node 4 (space-separated): 5  
Enter node 5: 5  
Enter neighbors of node 5 (space-separated): 4  
Enter the start node: 1  
DFS Traversal Order:  
['1', '2', '4', '5', '3']  
  
BFS Traversal Order:  
['1', '2', '3', '4', '5']