Problem 1 : Given an undirected graph, return a vector of all nodes by traversing the graph using depth-first search (DFS).

```
def dfs(graph, start_node, visited, traversal_order):
    visited[start_node] = True
    traversal_order.append(start_node)

    for neighbor in graph[start_node]:
        if not visited[neighbor]:
            dfs(graph, neighbor, visited, traversal_order)

def dfs_traversal(graph):
    num_nodes = len(graph)
    visited = [False] * num_nodes
    traversal_order = []

    for node in range(num_nodes):
        if not visited[node]:
            dfs(graph, node, visited, traversal_order)

    return traversal_order

example_graph = {
    0: [1, 2],
    1: [0, 2, 3],
    2: [0, 1, 3],
    3: [1, 2, 4],
    4: [3, 5],
    5: [4],
    6: [7],
    7: [6]
}
```

```
traversal_result = dfs_traversal(example_graph)

print("DFS Traversal Order:", traversal_result)
```