Practice Question

You are given a directed acyclic unweighted graph where each vertex represents a task ID, and an edge A→B indicates that task A must be completed before task B.
The graph is represented as follows:

Tasks: [T1,T2,T3,T4,T5,T6]

Prerequisites:

- T1→T3
- T2→T3
- T3→T4
- T3→T5
- T4→T6

Find a **linearized sorted order** of the tasks using the **topological sort algorithm**. Provide the pseudo-code and the steps of the algorithm.

Pseudo code:

```
vector<string> topologicalSort(vector<vector<string>>& graph,
vector<string>& vertices) {
    unordered_map<string, int> inDegree;
    unordered_map<string, vector<string>> adjList;
    vector<string> topoOrder;

    // Build adjacency list and calculate in-degree
    for (auto& edge : graph) {
        adjList[edge[0]].push_back(edge[1]);
        inDegree[edge[1]]++;
        if (inDegree.find(edge[0]) == inDegree.end())
inDegree[edge[0]] = 0;
    }

    // Initialize the queue with vertices of in-degree 0
    queue<string> q;
    for (auto& v : vertices) {
        if (inDegree[v] == 0) q.push(v);
    }

    // Process the queue
    while (!q.empty()) {
        string curr = q.front();
        q.pop();
        topoOrder.push_back(curr);
```

```
        for (string neighbor : adjList[curr]) {
            inDegree[neighbor]--;
            if (inDegree[neighbor] == 0) q.push(neighbor);
        }
    }

    return topoOrder;
}
```