Topological Sort – Kahn’s Algorithm

```
// Function to perform Topological Sort using Kahn's Algorithm
vector<int> topologicalSortKahn(int numVertices, vector<vector<int>>& adjList) {
    vector<int> inDegree(numVertices, 0); // Array to store in-degrees
    vector<int> topoOrder;               // Vector to store the topological order

    // Step 1: Calculate in-degree for each vertex
    for (int u = 0; u < numVertices; u++) {
        for (int v : adjList[u]) {
            inDegree[v]++;
        }
    }

    // Step 2: Enqueue all vertices with in-degree 0
    queue<int> q;
    for (int i = 0; i < numVertices; i++) {
        if (inDegree[i] == 0) {
            q.push(i);
        }
    }
```

```
    // Step 3: Perform BFS
    while (!q.empty()) {
        int current = q.front();
        q.pop();
        topoOrder.push_back(current);

        for (int neighbor : adjList[current]) {
            inDegree[neighbor]--;
            if (inDegree[neighbor] == 0) {
                q.push(neighbor);
            }
        }
    }

    // Step 4: Check for cycles
    if (topoOrder.size() != numVertices) {
        cout << "Graph contains a cycle. Topological sort
not possible." << endl;
        return {};
    }

    return topoOrder;
```

```
}

int main() {
    int numVertices = 6; // Example graph with 6 vertices
    vector<vector<int>> adjList = {
        {2, 3},  // Vertex 0 points to 2, 3
        {3, 4},  // Vertex 1 points to 3, 4
        {},      // Vertex 2 points to no one
        {5},     // Vertex 3 points to 5
        {5},     // Vertex 4 points to 5
        {}       // Vertex 5 points to no one
    };

    // Perform Topological Sort
    vector<int> result = topologicalSortKahn(numVertices,
adjList);
```