Kahn in C++

```
#include <bits/stdc++.h>
using namespace std;

class Solution {
public:
   //Function to return list containing vertices in
Topological order.
   vector<int> topoSort(int V, vector<int> adj[])
   {
      int indegree[V] = {0};
      for (int i = 0; i < V; i++) {
         for (auto it : adj[i]) {
            indegree[it]++;
         }
      }

      queue<int> q;
      for (int i = 0; i < V; i++) {
         if (indegree[i] == 0) {
            q.push(i);
         }
      }
      vector<int> topo;
      while (!q.empty()) {
         int node = q.front();
         q.pop();
         topo.push_back(node);
         // node is in your topo sort
         // so please remove it from the indegree

         for (auto it : adj[node]) {
            indegree[it]--;
            if (indegree[it] == 0) q.push(it);
         }
      }

      return topo;
   }
};

int main() {

   //V = 6;
   vector<int> adj[6] = {{}, {}, {3}, {1}, {0, 1}, {0, 2}};
   int V = 6;
   Solution obj;
   vector<int> ans = obj.topoSort(V, adj);

   for (auto node : ans) {
      cout << node << " ";
   }
   cout << endl;

   return 0;
}
```

Input Graph (Adjacency List)

```
vector<int> adj[6] = {
    {},       // 0
    {},       // 1
    {3},      // 2 → 3
    {1},      // 3 → 1
    {0, 1},   // 4 → 0, 1
    {0, 2}    // 5 → 0, 2
};
```

🧮 Step 1: Calculate In-Degree of Each Node

Node	Incoming Edges from	In-degree
0	4, 5	2
1	3, 4	2
2	5	1
3	2	1
4	-	0
5	-	0

➡ Initial indegree[] = {2, 2, 1, 1, 0, 0}

📥 Step 2: Enqueue All Nodes With In-degree = 0

Initial Queue: q = [4, 5]

🔁 Step 3: BFS Loop & Topological Sorting

Iteration	Node Popped	Topo List	Decrease In-degree	Queue after Push
1	4	[4]	0→1, 1→1	[5]
2	5	[4, 5]	0→0 ✅, 2→0 ✅	[0, 2]
3	0	[4, 5, 0]	-	[2]
4	2	[4, 5, 0, 2]	3→0 ✅	[3]
5	3	[4, 5, 0, 2,	1→0 ✅	[1]

Iteration	Node Popped	Topo List	Decrease In-degree	Queue after Push
		3]		
6	1	[4, 5, 0, 2, 3, 1]	-	[] (done)

✅ Final Output

Topological Order = [4, 5, 0, 2, 3, 1]

🧠 Summary Table

Node	Final In-degree	Status
0	0	Printed
1	0	Printed
2	0	Printed
3	0	Printed
4	0	Printed
5	0	Printed

Output:-
4 5 0 2 3 1