Reverse directed graph in C++

```
#include <iostream>
#include <vector>
using namespace std;

class ReverseDirectedGraph {
public:
    static vector<vector<int>>
reverseDirectedGraph(const vector<vector<int>>& adj,
int V) {
        vector<vector<int>> reversedAdj(V + 1);

        for (int i = 0; i <= V; ++i) {
            for (int j : adj[i]) {
                reversedAdj[j].push_back(i);
            }
        }

        return reversedAdj;
    }

    static void printGraph(const vector<vector<int>>&
graph, int V) {
        for (int i = 1; i <= V; ++i) {
            for (int j : graph[i]) {
                cout << i << " -> " << j << endl;
            }
        }
    }
};

int main() {
    int V = 5;
    vector<vector<int>> adj(V + 1);

    adj[1].push_back(3);
    adj[1].push_back(2);
    adj[3].push_back(4);
    adj[4].push_back(5);

    vector<vector<int>> reversedAdj =
ReverseDirectedGraph::reverseDirectedGraph(adj, V);

    cout << "Reversed Graph:" << endl;
    ReverseDirectedGraph::printGraph(reversedAdj, V);

    return 0;
}
```

Original Input Graph (Adjacency List)

We have a **directed graph** with 5 vertices (V = 5):

Vertex	Edges
1	→ 3, → 2
2	—
3	→ 4
4	→ 5
5	—

Graphically:

1 → 2
↓
3 → 4 → 5

🔁 Dry Run Table: reverseDirectedGraph(adj, V)

This function creates a reversed adjacency list where **every edge u → v becomes v → u**.

i (Source Node)	j (adj[i])	reversedAdj[j] After Insertion
1	3	reversedAdj[3] = {1}
1	2	reversedAdj[2] = {1}
3	4	reversedAdj[4] = {3}
4	5	reversedAdj[5] = {4}

📥 Final reversedAdj Table

Vertex	reversedAdj[vertex] (Incoming Edges)
1	—
2	1
3	1
4	3
5	4

🖨️ Output of printGraph(reversedAdj, V)

This prints **destination → source** (reversed):

	2 -> 1 3 -> 1 4 -> 3 5 -> 4
Output:- Reversed Graph: 2 -> 1 3 -> 1 4 -> 3 5 -> 4	