// Biểu diễn đồ thị bằng Danh sách kề dùng Vector trong STL

#include <iostream>

#include <vector>

using namespace std;

// Data structure to store a graph edge

struct Edge {

int src, dest;

};

// A class to represent a graph object

class Graph

{

public:

// a vector of vectors to represent an adjacency list

vector<vector<int>> adjList;

// Graph Constructor

Graph(vector<Edge> const &edges, int n)

{

// resize the vector to hold `n` elements of type `vector<int>`

adjList.resize(n);

// add edges to the directed graph

for (auto &edge: edges)

{

// insert at the end

adjList[edge.src].push\_back(edge.dest);

// uncomment the following code for undirected graph

// adjList[edge.dest].push\_back(edge.src);

}

}

};

// Function to print adjacency list representation of a graph

void printGraph(Graph const &graph, int n)

{

for (int i = 0; i < n; i++)

{

// print the current vertex number

cout << i << " ——> ";

// print all neighboring vertices of a vertex `i`

for (int v: graph.adjList[i]) {

cout << v << " ";

}

cout << endl;

}

}

// Graph Implementation using STL

int main()

{

// vector of graph edges as per the above diagram.

// Please note that the initialization vector in the below format will

// work fine in C++11, C++14, C++17 but will fail in C++98.

vector<Edge> edges =

{

{0, 1}, {1, 2}, {2, 0}, {2, 1}, {3, 2}, {4, 5}, {5, 4}

};

// total number of nodes in the graph (labelled from 0 to 5)

int n = 6;

// construct graph

Graph graph(edges, n);

// print adjacency list representation of a graph

printGraph(graph, n);

return 0;

}