#include <bits/stdc++.h>

using namespace std;

#define V 5

void printSolution(const vector<int>& path);

bool isSafe(int v, const vector<vector<bool>>& graph, const vector<int>& path, int pos) {

if (graph[path[pos - 1]][v] == 0)

return false;

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

if (path[i] == v)

return false;

return true;

}

bool hamCycleUtil(const vector<vector<bool>>& graph, vector<int>& path, int pos) {

if (pos == V) {

if (graph[path[pos - 1]][path[0]] == 1)

return true;

else

return false;

}

for (int v = 1; v < V; v++) {

if (isSafe(v, graph, path, pos)) {

path[pos] = v;

if (hamCycleUtil(graph, path, pos + 1))

return true;

path[pos] = -1;

}

}

return false;

}

bool hamCycle(const vector<vector<bool>>& graph) {

vector<int> path(V, -1);

path[0] = 0;

if (!hamCycleUtil(graph, path, 1)) {

cout << "\nSolution does not exist";

return false;

}

printSolution(path);

return true;

}

void printSolution(const vector<int>& path) {

cout << "Solution Exists: Following is one Hamiltonian Cycle \n";

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

cout << path[i] << " ";

cout << path[0] << " ";

cout << endl;

}

int main() {

vector<vector<bool>> graph(V, vector<bool>(V));

cout << "Enter the adjacency matrix of the graph (size 5x5):\n";

for (int i = 0; i < V; i++) {

for (int j = 0; j < V; j++) {

cin >> graph[i][j];

}

}

hamCycle(graph);

return 0;

}