**Maze solver**

#include <iostream>

#include <vector>

using namespace std;

// Directions for moving in the maze (right, down, left, up)

const int dx[] = {0, 1, 0, -1};

const int dy[] = {1, 0, -1, 0};

// Function to check if a cell is within the maze bounds and is not a wall

bool isSafe(const vector<vector<int>>& maze, vector<vector<bool>>& visited, int x, int y) {

    int n = maze.size();

    return (x >= 0 && x < n && y >= 0 && y < n && maze[x][y] == 0 && !visited[x][y]);

}

// DFS function to find the path

bool solveMaze(vector<vector<int>>& maze, vector<vector<bool>>& visited, int x, int y, vector<pair<int, int>>& path) {

    int n = maze.size();

    // Check if we've reached the end

    if (x == n-1 && y == n-1) {

        path.push\_back({x, y});

        return true;

    }

    // Check if this cell is safe to visit

    if (isSafe(maze, visited, x, y)) {

        visited[x][y] = true;

        path.push\_back({x, y});

        // Explore neighbors (right, down, left, up)

        for (int i = 0; i < 4; ++i) {

            int newX = x + dx[i];

            int newY = y + dy[i];

            if (solveMaze(maze, visited, newX, newY, path)) {

                return true;

            }

        }

        // If no path found, backtrack

        path.pop\_back();

        visited[x][y] = false;

        return false;

    }

    return false;

}

int main() {

    int n;

    cout << "Enter the size of the maze: ";

    cin >> n;

    vector<vector<int>> maze(n, vector<int>(n));

    cout << "Enter the maze (" << n\*n << " values of 0 or 1):\n";

    for (int i = 0; i < n; ++i) {

        for (int j = 0; j < n; ++j) {

            cin >> maze[i][j];

        }

    }

    vector<vector<bool>> visited(n, vector<bool>(n, false));

    vector<pair<int, int>> path;

    if (solveMaze(maze, visited, 0, 0, path)) {

        cout << "Path found:\n";

        for (const auto& p : path) {

            cout << "(" << p.first << ", " << p.second << ") ";

        }

        cout << endl;

    } else {

        cout << "No path found.\n";

    }

    return 0;

}