// 200-Number\_of\_Islands.cpp : BFS //

#include <iostream>

#include <queue>

using namespace std;

class Solution {

public:

int numIslands(vector<vector<char>>& grid) {

int nr = grid.size(); /\* row \*/

if (!nr) return 0;

int nc = grid[0].size(); /\* column \*/

int num\_islands = 0;

for (int r = 0; r < nr; ++r) {

for (int c = 0; c < nc; ++c) {

if (grid[r][c] == '1') {

++num\_islands;

grid[r][c] = '0'; // mark as visited

queue<pair<int, int>> Q;

Q.push({ r, c });

while (!Q.empty()) {

auto cur = Q.front();

Q.pop();

int row = cur.first, col = cur.second;

/\* check all four edges \*/

if (row + 1 < nr && grid[row + 1][col] == '1') {

Q.push({ row + 1, col }); grid[row + 1][col] = '0';

}

if (col + 1 < nc && grid[row][col + 1] == '1') {

Q.push({ row, col + 1 }); grid[row][col + 1] = '0';

}

if (row - 1 >= 0 && grid[row - 1][col] == '1') {

Q.push({ row - 1, col }); grid[row - 1][col] = '0';

}

if (col - 1 >= 0 && grid[row][col - 1] == '1') {

Q.push({ row, col - 1 }); grid[row][col - 1] = '0';

}

}

}

}

}

return num\_islands;

}

};

int main()

{

Solution sol;

vector < vector<char>> vect = { {'1','1','1','1','0'}, {'1','1','0','1','0'}, {'1','1','0','0','0'}, {'0','0','0','0','0'} };

vector < vector<char>> vect2 = { {'1','1','0','0','0'}, {'1','1','0','0','0'}, {'0','0','1','0','0' }, {'0','0','0','1','1' } };

cout << sol.numIslands(vect) << endl;

cout << sol.numIslands(vect2) << endl;

}