```
/* FILE: solver.cpp   last change: 10-Feb-2015   author: Romeo Rizzi
 * a linear solver for problem "connect"
 */

#define NDEBUG   // NDEBUG definita nella versione che consegno
#include <cassert>
#ifndef NDEBUG
#  include <iostream>  // uso di cin e cout non consentito in versione finale
#endif
#include <fstream>

using namespace std;

const int MAX_M = 1000; const int MAX_N = 1000;
int p; int B[MAX_M+2][MAX_N+2]; // p pebbles are placed on a board B
int m, n; // actual dimensions of the board. Num rows and cols.

int num_con = 0, num_cri = 0; // the two answers
int seen[MAX_M+2][MAX_N+2], back[MAX_M+2][MAX_N+2];
int open[MAX_M+2][MAX_N+2], t = 0;
int cn[MAX_M+2][MAX_N+2]; // characteristic vector of the critical nodes

void BFS(int i, int j, int i_dad, int j_dad) {
  seen[i][j] = 1; back[i][j] = open[i][j] = t++;
  int num_children = 0;
  for(int di = -1; di <= 1; di++)
    for(int dj = -1; dj <= 1; dj++)
      if( di*dj == 0 )
      if( B[i+di][j+dj] ) {
        if( !seen[i+di][j+dj] ) {
          num_children++;
          BFS( i+di, j+dj, i, j );
          if( back[i+di][j+dj] < back[i][j] ) back[i][j] = back[i+di][j+dj];
          if( back[i+di][j+dj] >= open[i][j] )
            if( (i != i_dad) || (j != j_dad) ) // if I am not root
               cn[i][j] = 1;
        }
        else
          if( (i+di != i_dad) || (j+dj != j_dad) )
            if( open[i+di][j+dj] < back[i][j] ) back[i][j] = open[i+di][j+dj];
      }
  if( (i == i_dad) && (j == j_dad) ) // if I am root
    if( num_children > 1 ) cn[i][j] = 1;
  if( cn[i][j] )  num_cri++;
}

int main() {
  ifstream fin("input.txt"); assert( fin );
  fin >> m >> n >> p;
  int x, y;
  for(int i = 0; i < p; i++) {
    fin >> x >> y;
    B[x][y] = 1;
  }
  fin.close();

  for(int i = 1; i <= m; i++)
    for(int j = 1; j <= n; j++)
      if( B[i][j] )
        if( !seen[i][j] ) {
          num_con++;
          BFS(i,j, i,j);
        }

  ofstream fout("output.txt");
  fout << num_con << endl << num_cri << endl;
  fout.close();
  return 0;
}
```