COP 3223 Program #10: Minesweeper Setup (minesweeper.c)

Objective

To give students practice in writing functions and calling those functions to perform more complex tasks.

The Problem: Minesweeper Set Up

For this week you will only write one program. **Your program will read its input from the file, "mine.txt" and write output to the screen.**

This file will have information about multiple minesweeper boards. In particular, it will tell you the location of each mine on the board for your program to evaluate. Your program must read in this information and then print out a version of the board that has a '*' where each bomb is located and a number indicating the number of adjacent bombs for each non-bomb location.

The format of the input file is as follows:

The first line of the input file contains a single positive integer, *n*, representing the number of minesweeper boards to construct.

The first line of each board contains a single positive integer, *m*, representing the number of bombs for that board. The following *m* lines will contain two non-negative integers in between 0 and 7, inclusive, representing the row and column, respectively, of the location of a bomb. No two of these *m* lines will be identical.

For each board, output to the screen, the following header line:

```
Board #k:
```

where k is the number of the board, starting with 1.

Follow this with the board printed out over 8 lines, with a space in between each entry.

Separate the output for each case with a blank line.

Hints

Store the board in an 8 x 8 integer array. Store the number 9 where bombs are supposed to be located. When printing the board, use an if statement to screen for bombs. Both 8 and 9 should be represented by symbolic constants in this program. Watch out for array out of bounds issues by using if statements to screen for illegal locations. One neat idea to fill the board would be to initialize all non-bomb squares to 0 and then add one to each square adjacent to each bomb (except for other bombs!) The other alternative idea is to go to each non-bomb square and look at each location adjacent to it. Also, an inbounds function that takes in an x and y coordinate and returns whether or not it's inbounds may be helpful.

Sample Input

```
2
3
0 0
2 1
7 7
10
0 0
0 1
0 2
0 3
0 4
0 5
0 6
0 7
7 0
7 1
```

Corresponding Sample Output

```
Board #1:
* 1 0 0 0 0 0 0
2 2 1 0 0 0 0 0
1 * 1 0 0 0 0 0
1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 *

Board #2:
* * * * * * * *
2 3 3 3 3 3 3 2
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
2 2 1 0 0 0 0 0
* * 1 0 0 0 0 0
```

Restrictions

Although you may use other compilers, your program must compile and run using gcc in Code::Blocks. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments throughout your code, when appropriate. If you have any questions about this, please see a TA. **Your program must read its input from mine.txt.**

Deliverables

A single source file named *minesweeper.c* turned in through WebCourses.