Homework Project #4 – Minesweeper part 2

**Assigned:** June 15h, 2015

**Due:** July 6th (11:59pm)

**Submission Format:** You will submit a soft copy of your C program solution. Name the submission file HW4.c. This copy will be submitted through Canvas (http://lss.at.ufl.edu/).

**Dependency on homework #3:** As mentioned in class this is the second part of a project we started with homework #3. In case you don’t have a working solution for homework #3, the instructor’s solution will be posted on the class website as soon as everyone has submitted their homework #3.

**Assignment:**

Our projects will lead us through the creation of a text-based Minesweeper game. Each project will be adding new functionality to the game. If you have never played Minesweeper, you can find an online version here! The rules and game play are easy, you can read the overview in wiki! In Homework #4, we will continue the assumption that the board is always 9 by 9 (the beginner size in most implementations). Spoiler alert, this assumption will no longer be true for homework #5.

In this homework, we will:

- Allow interactive gameplay.
- Detect the losing condition. (The winning condition will be part of HW #5, but you are welcome to look into it. None of the test cases will involve the user winning, so your output for the win condition doesn’t matter).
- **For extra credit:** add color to the board

**Input**

To begin the game, your program will receive the same two integer numbers as HW #3. As a reminder:

1. A **seed** for the random number generator. You are expected to call srand() with this seed before assigning mines.
2. A number of **mines** that will be placed in the board. You are expected to validate that the number of mines is positive, greater than 0 and less than the number of tiles (81).

After those two, your program will print the full board completely concealed (this is not the solution). Following that the user will be able to input as many **plays** as she needs until the **play** results on opening a mine. This is the losing condition.
A play consists of one single character identifying the action followed by a pair (char, int) identifying the tile the action is affecting. In a scanf you would read it as "%c(%c,%d)". You will need to validate that the tile exists and that the actions is among the implemented ones.

The possible actions you need to implement are:
- **O**: opens tile (equivalent of left click in most implementations)
- **M**: marks a tile as a mine, if the tile was marked as mine it unmarks it (equivalent to right click in most implementations).
- **Q**: marks a tile as a question mark. Question marks are helpful to the user to check tiles that could be a mine.

Notice that trying to Open a tile marked as a mine or a question mark is an invalid play!

**Output**

1. After the seed and # of mines you will output an unplayed board.
2. After each valid play, you will output the current state of the board.
3. After each invalid play, you will print a helpful menu with the valid plays.

If the play resulted in the player losing, the game ends after the end board is printed.

**Opening a tile:**

There are three cases for what happens when the user opens a tile:

1. If it is a mine: the user loses! Print losing board, then end game.
2. If it is a number: only the tile gets open. The user gets to play again.
3. If it is an ' _ ' (that is a 0). This opens that all adjacent tiles that are not mines (recursively). Your implementation doesn’t have to be recursive, but it does need to open all the adjacent tiles. The user gets to play again.
Printing the state of the board:

The very first line of the board starts with two blank spaces, then a letter followed by a space for each column starting at ‘A’ (until ‘I’ since we have 9 columns).

Then for each row, start the line with the number of the row (starting at 1), followed by a space. Then the symbols for each tile on that row separated by spaces. Each tile can contain one of 4.5 options:

1. An unopened tile: use '*'
2. A tile marked as a mine: use '!
3. A tile marked as a possible mine/question mark: use '?'
4. An opened tile: opened tiles follow the rules for the solution:
   a. '  ' If the tile has no surrounding mines.
   b. A number representing the number of mines surrounding the tile.

Printing the losing board:

The very first line of the board starts with two blank spaces, then a letter followed by a space for each column starting at ‘A’ (until ‘I’ since we have 9 columns).

Then for each row, start the line with the number of the row (starting at 1), followed by a space. Then the symbols for each tile on that row separated by spaces. Each tile can contain one of 4.5 options:

1. If the tile is a mine in the solution:
   a. If it was marked as mine by the user: use '!
   b. If the user played this mine to lose use: '#'
   c. Else use '@'
2. If the tile is not a mine in the solution:
   a. If it is marked as a mine by the user: use 'x'
   b. If it has been open by the user: follow rules for open tiles (',_' or number)
   c. If it is closed: use '*'

Adding color (extra credit!)

You are free to be creative about the coloring scheme. Notice that there are colors already defined in the minesweeper game, but we don’t have enough colors in the console to replicate that. In order to get the extra credit the minimum requirement is that the following have different colors:

- Mines
- Opened tiles (numbers and _)
- Unopened tiles and the border (meaning *, A-I for column headers and 1-9 for row headers, can all use the same color)

Examples are a bit large and will be posted 6/17.
<table>
<thead>
<tr>
<th>Section</th>
<th>Grading criterion</th>
<th>Point value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program functionality (75%)</td>
<td>Allow multiple plays</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Validate plays/print message</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Correctly implementing all three actions</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>Correctly printing current state and losing board</td>
<td>10%</td>
</tr>
<tr>
<td>Good programming practices (10%)</td>
<td>Beginning your file with a comment that includes: your name, the name of the class, your instructor’s name and a brief explanation of what the program does. (Include any issues that you were unable to solve in this comment)</td>
<td>3%</td>
</tr>
<tr>
<td></td>
<td>Use meaningful variable names</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Use good tabulation</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Provide meaningful comments at least for your strategy on computing the solution numbers</td>
<td>3%</td>
</tr>
<tr>
<td>Compilation (no errors – no warnings)</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>Extra points – Coloring</td>
<td></td>
<td>10%</td>
</tr>
</tbody>
</table>