Homework Project #3 – Minesweeper part 1

Assigned: June 1st, 2015
Due: June 15th (11:59pm)

Submission Format: You will submit a soft copy of your C program solution. Name the submission file HW3.c. This copy will be submitted through Canvas (http://lss.at.ufl.edu/).

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 #3, we will assume the board is always 9 by 9 (the beginner size in most implementations).

In this homework, we will:
- Generate the board, including the random assignment of the mines.
- Compute the solution to the board. Meaning the numbers for each tile.
- Print the solution.

Input
Your program will receive two integer numbers as input in this order:
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).

Output
Once those two numbers are read, you will:
1. Assign the positions of the mines.
2. Compute the solution.
3. Print the solution.

Assigning the mines
In order to guarantee that the solutions are deterministic (i.e.: that everyone gets the same board given the same parameters), this will be the strategy to follow to assign mines:
1. Compute a random number between 0 and 8 for the ROW
2. Compute a random number between 0 and 8 for the COLUMN
3. Check if that tile (ROW, COLUMN) has a mine assigned. If it doesn’t have a mine, assign the mine. If it does, don’t do anything.

4. Repeat the process until you have assigned the number of mines the user gave as input.

**Computing the solution**

This is the core functionality for this homework. For each tile that does not have a mine, you will need to compute how many mines are around it. As an example, mines in any one of the blue squares of the tile below count as neighbors of the orange tile. Since all of them have a mine (@), then the number assigned is 8.

```
  @  @  @
  @   8 @
  @  @  @
```

As a second example, if only 3 blue neighbors had mines the number in the orange cell would be 3.

```
  @  @  @
  @  3 @
  @   @
```

Computing the numbers is up to you, the strategy you use needs to be commented. Notice that edge cases need to be carefully dealt with. Failure to consider those cases will often result in “Segmentation fault” errors.

**Printing the solution**

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 3 options:

a) A **mine**: use an ‘@’ for tiles containing a mine.

b) A **number**: if the tile doesn’t have a mine but has at least one neighbor with a mine, then print the number of neighbors that do have mines (computed in the solution).

c) An **underscore**: if a tile doesn’t have a mine, and doesn’t have neighbors containing mines, print an ‘_’
Example:

Please input your seed: 5000
Please input the number of mines: 10

```
  A B C D E F G H I
1 _ _ _ _ _ _ 1 @ 1
2 _ _ _ _ _ _ 1 1 1
3 1 1 _ _ _ _ 1 1 1
4 @ 1 _ _ _ _ 1 @ 1
5 2 2 1 2 2 1 1 2 2
6 @ 2 2 @ @ 1 _ 1 @
7 1 2 @ 3 2 1 1 2 2
8 1 2 1 1 _ _ 1 @ 1
9 @ 1 _ _ _ _ 1 1 1
```

<table>
<thead>
<tr>
<th>Section</th>
<th>Grading criterion</th>
<th>Point value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program functionality (75%)</strong></td>
<td>Reading the number of mines and validating it is less than the number of tiles in the board</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Following the mine assignment protocol</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Computing the solution correctly</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>Printing the solution correctly</td>
<td>10%</td>
</tr>
<tr>
<td><strong>Good programming practices (10%)</strong></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><strong>Compilation (no errors – no warnings)</strong></td>
<td></td>
<td>15%</td>
</tr>
</tbody>
</table>