Minesweeper

"C is quirky, flawed, and an enormous success."

Dennis M. Ritchie

Please read the whole document before starting

1 Background

A good start would to read the overview the Minesweeper Wiki page. Then you can watch a quick video of it on YouTube. And if you want, play a quick game of it online.

Or you can just look at a quick summary of the rules of Minesweeper:

- Each turn a user picks a square, (x, y) coordinates.
- If you hit a mine, you lose.
- Else the user picks another square.
- If you clear out all the squares besides the mine tiles? You Win!

2 Specifications

1. Minesweeper is usually a graphical game, but for simplicity ours will be text-based. The program will also offer no prompts; you have to know what it expects. This makes it somewhat user unfriendly, but an output uncluttered by prompts is easier to check visually and with differencing tools. All lines must have a final return, including the last line.
2. The board dimensions are 20×20, with 0-based indexing
3. We'll manually specify the mines' location, making the game less interesting, but easier to test. Your program thus starts by reading a mine count (without prompt) giving the number of mines to place on the board.
    - If mine count is negative, display the line: **Number of mines cannot be negative.** and exit.
4. For each mine, the program then reads a space-separated row and column giving the location of the mine.
    - If the coordinates are out of bounds, or if they are the same as an already-placed mine, display **(row, col) is not a valid position for a mine.** where row and col are the numbers entered. Omit the bad mine, and continue reading mine locations
    - If you reach EOF before reading enough coordinate pairs, display: **Not enough mines placed on the board.** and exit.

5. Once all of the mine positions have been set, display the board in the following format:

```
Initial Board:
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
```

- There are no blank lines between rows, but there are spaces between each column, along with a space and return at the end of each row.

6. After displaying the board for the first time, read in pairs of coordinates and reveal the tile at that location, plus any further tiles with no mines as neighbors, displaying the entire board after every guess.

   - A * stands for a tile that has yet to be revealed.
   - An X stands for a mine.
   - A digit indicates the number of mines surrounding that tile. Use a '0' instead of the usual blank space for tiles with no mines around them.
   - Keep a count of the the round the user is on, numbering from 0.

- For example, the following board might be shown after the user inputs "0 0" as their first guess:

```
Round:  0
1 * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
```

- As you know from observing an example of the game, if an exposed cell has 0 neighboring mines, all unexposed neighbor cells are automatically exposed, since they're obviously "safe". And this process continues in a wave or "flood fill" through as many 0-neighbor cells as are connected to the original, until non 0-neighbor cells are reached. If (0, 0) had no adjacent mines, and there was a particularly large flood fil1 of 0-mine locations, we might see:

```
Round:  0
0 0 2 * * * * * 1 0 0 0 0 0 0 0 0 0 0
0 0 0 2 * * * * * 1 0 0 0 0 0 0 0 0 1 1
0 0 0 1 1 1 2 * 3 1 0 0 0 0 0 0 0 0 1 *
1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1
* * 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
* * * 1 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 0 0 0 0 0 0 0 0 0 0 0
* * 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
* * 1 0 0 0 0 0 0 0 1 * 1 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 1 2 2 1 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 0
0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 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 1 * 1 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 3 * *
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 * * * *
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 * * * *
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 * * * *
0 0 0 1 * 1 0 0 1 1 1 0 0 0 0 1 * * * *
0 0 0 1 * 1 0 0 1 * 1 0 0 0 0 1 * * * *
```

- If the numbers supplied are out of bounds on the board, display: **(x, y) is not a valid position.** with a return at the end of the statement, and where x and y are the numbers entered. Continue the game.

- If the coordinates are valid but the spot has already been revealed, display: **Spot already revealed, pick another spot** with a return at the end of the statement. Continue the game.

7. After each guess, check to see if the game is over (if the player has lost or all non-mine spots have been revealed).

   - If the user clicked on a mine, display the following message: **You Lose...** followed by the entire board with all spots revealed.

```
You Lose . . .
0 0 0 2 X X X 3 2 1 0 0 0 0 0 0 0 0 0 0
0 0 0 2 X 4 4 X X 1 0 0 0 0 0 0 0 0 1 1
0 0 0 1 1 1 2 X 3 1 0 0 0 0 0 0 0 0 1 X
1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1
1 X 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 X 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
1 X 1 0 0 0 0 0 0 1 X 1 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 1 2 2 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 X 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 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 1 X 1 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 3 X 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 X X 2 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 1 0
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 1 X 1 0 0 1 1 1 0 0 0 0 1 X 1 0 0
0 0 0 1 1 1 0 0 1 X 1 0 0 0 0 1 1 1 0 0
```

   - If the player has won, display: **You Win!!!!** followed by the entire board.
   - If input reaches EOF before the game has ended, display: **No more input. Game Over.** and exit.

3 Coding and Submission

You must use LearningIDE, at LearningIDE, to do all work on this assignment, starting with the first keystroke. This site will keep a record of your work on the assignment, and use of it is **required** for credit on the assignment.

It is instructionally useful for us to see how you reason about the assignment. But, you wont be graded on your intermediate work, just on the final result you submit to the course grading system.

The site is quite easy to use, offering a simple editing UI, and basic compilation and running facilities. It also has a new beta-release C debugger that you are welcome to try out. Documentation is available here.

When you create your LearningIDE project, call it **MineSweeper**, and base it on the assignment of that same name from instructor **awang24@calpoly.edu** (Enter those two things in the Project creation dialog). This will set you up with a properly named C file, and the set of test cases you must pass to complete the assignment.

When you have passed all the supplied test cases (all show green), and you've made no edits since getting all of them green, then you can submit your project via the Submit option on the Project menu. If you submit is approved, you're done. It's not necessary to have the hand-run test show green; just the canned

ones. **Do not modify any of the supplied test cases**.

The due date is posted on the projects page. For late submissions, it is 5% off per day late.