MineSweeper

Windows versions have been including the „Minesweeper“ game for quite a while now. The goal is to survive the detection of all bombs randomly hidden in a mine field. In our text-based version, the field is a simple 9 x 9-matrix with (initially) all cells undisclosed. At the beginning, the user is asked for the number of bombs to be randomly placed on the field. Before each move, the current state of the field (including number of bombs in total/marked) is displayed:

```
************************
* M I N E S W E E P E R *
************************
Enter number of bombs (1-80): \stdin{20}
Bombs: 20 Marked: 0
1 2 3 4 5 6 7 8 9
1 * * * * * * * * *
2 * * * * * * * * *
3 * * * * * * * * *
4 * * * * * * * * *
5 * * * * * * * * *
6 * * * * * * * * *
7 * * * * * * * * *
8 * * * * * * * * *
9 * * * * * * * * *
```

Then, the user is asked to enter the coordinates of a cell, and additionally to decide whether a cell should be inspected (at the risk of finding a bomb, in which case the user looses) or just marked as a bomb (if the user thinks the cell hides one). The format for this is yxc, where y denotes the row, x the column and c the command: + for inspecting a cell, and – for marking/unmarking a cell:

```
Your move: \stdin{23+}
Bombs: 20 Marked: 0
1 2 3 4 5 6 7 8 9
1 * * * * * * * * *
2 * * 4 * * * * * *
3 * * * * * * * * *
4 * * * * * * * * *
5 * * * * * * * * *
6 * * * * * * * * *
7 * * * * * * * * *
8 * * * * * * * * *
9 * * * * * * * * *

Your move: \stdin{31-}
Bombs: 20 Marked: 1
1 2 3 4 5 6 7 8 9
```

generated by haroopad

```
1 * * * * * * * * *
2 * * 4 * * * * * *
3 B * * * * * * * *
4 * * * * * * * * *
5 * * * * * * * * *
6 * * * * * * * * *
7 * * * * * * * * *
8 * * * * * * * * *
9 * * * * * * * * *
```

If an inspected cell does not contain a bomb, the cell shows the number of bombs in the direct 8-neighborhood (e.g., for cell (1,1), the cells (1,2), (2,2) and (2,1)). The user wins if all cells but the ones with bombs have been opened/inspected. After the end of a game, the user is asked if he/she wants to play again. In the following example, Cell (9,1) contains a bomb:

```
Your move: \stdin{91+}
BUMM! You're dead!
Do you want to play again (y/n)? \stdin{n}
Coward!
```

Hints

- To simplify things, declare globally: a 9x9 integer array for the initial bomb field, a 9x9 character array for the screen shown to the user, and count variables for the total amount of bombs, the number of hidden (not yet inspected) cells, and the number of cells marked by the user as hiding a bomb.
- Write a function `void init_game(void)` that
  - prints out a start message,
  - initializes the bomb field with 0,
  - asks for the number of bombs,
  - sets the global variables (number of bombs, etc.),
  - for each bomb to be placed, seeks random coordinates and sets the corresponding field to -1 (Take care that you are not placing a bomb on place where you already put one before!),
  - for each cell, computes the number of bombs in the direct neighborhood and sets the cells in the field array accordingly. For this, you might:
  - Write a function `int count(int i, int j)` that checks if the coordinates (i, j) are correct and returns 1 if the cell contains a bomb and 0 otherwise. Then you just need to call this function for each cell “around” an inspected cell ((x, y-1), (x, y+1), (x-1, y), …) and sum up the results.
- Write a function `void inspect(int i, int j)` that checks an empty cell (i, j) that does not contain a bomb and updates the screen array (either a number for the bombs in the neighborhood, or a space if the number is 0).
- The `main()` function should initialize the game, and then do the loop (display current screen, ask for user input, check whether the user find a bomb or update the screen array). If a user wins/looses, the user is asked for a new game.