**Final project – Minesweeper.**

Welcome to the final project. It is not mandatory. You are allowed one submission of the code (that must be operational) for stylistic + algorithmic comments. An additional final submission after correction of the comments is also allowed. Don’t forget to document your code (only before functions).

Minesweeper is a game that was bundled with windows until windows 8. A link to a site explaining the rules (and allowing you to play) is given here: https://minesweeper.online/

General instructions:

- No chatgpt/google searches to solve the problem. I’m sure chatgpt can write minesweeper, and I’m sure you can find implementations. This project is not mandatory, and I have no interest in checking chatgpt code.

- You are, however, allowed to google specific commands/errors you are having issues with.

- The suggestions for writing Minesweeper given in this file are not optimal – but out workshop does not focus on software architecture best practices, or runtime/space complexity.

- We provide suggested steps on how to approach writing the game, but you can write things in whatever order you want, as long as you end up with a working, playable, game.

- Your cost must be modular – meaning it must be separated into methods (in a way that makes sense).

- It is highly recommended to test each function after every step.

- It is surprisingly easy to tell when someone copied code from someone else. If you want to work together with someone else that’s fine (for those who it isn’t a mandatory submission), just please only one of you submit.

The game:

The program should receive as a command line arguments 2 integer inputs (the size of the grid) and one double input (a probability, between 0 and 1, of a tile being a mine). The user is repeatedly asked for 2 integer coordinates (one at a time), and what to do with the tile (0 for left-click, 1 for right click (flag)). The grid is then printed.

If a non-mine tile is chosen, it is replaced with a number signifying the number of mines adjacent to it. If a mine is clicked, the grid so far is printed, showing all unflagged mines, and the game ends.

Suggested steps:

**Step 1:**

Write a function that receives grid dimensions and a probability, and returns a 2d-array of ints with -1 for every mine, and 0 for every non-mine

**Step 2:**

Write a function that receives a grid of mines, changes every 0 to the number of adjacent mines, and returns it.

**Step 3:**

Write a function that receives a coordinate, and returns the value in that coordinate.

**Step 4:**

Write a function that receives a coordinate, and a 2d-array of int's of the current status of all tiles, “clicks” the value in it, and returns the 2d-array of the current status of all the tiles.

**Bonus step 4.1:**

If a clicked tile has no mine neighbors – automatically “click” all adjacent tiles (and keep going).

**Step 5:**

Write a function that receives a coordinate, and a 2d-array of ints of the current status of all tiles, “flags” the tile, and returns the 2d-array of the current status of all the tiles.

**Bonus step 5.1:**

also unflag a flagged tile with the same function.

Bonus Step: Allow clicking on a previously marked tile (equivalent to clicking all unmarked tiles surrounding it).

**Step 6:**

Write a function that receives the mine grid and the status grid, and prints the display the user should have

**Step 7:**

Write a function that receives the status and mine grid, and returns if the game is complete.

**Step 8:**

Read about the Scanner class (https://www.w3schools.com/java/java\_user\_input.asp).

Understand how to get user input for the coordinates. Write a matching function for getting user input (clicks/flags). Include handling for invalid inputs (flagging a marked tile, tile that doesn’t exist, and so on).

**Step 8:**

Connect everything.

Bonus Step – split to functions in more places that make sense – such as the user input.

Bonus Step – think about runtime. Are there are times you go over the entirety of a 2d grid (which takes a long time) and you don’t actually need to?

Checklist before submission:

* Can you play your game?
* Is the game making sense? Are the games’ rules being followed by the computer?
* Does it tell you if you lost, and stop the game correctly?
* Does it tell you if you win, and stop the game correctly?
* Can you break your game with type-correct input (so its fine if your game crashes when given a string when it expects an int. It shouldn’t crash if you try to access tile [7][7] in a 4x4 array, just tell the user and way for a different response.
* Is there repeated code?
* Is everything using conventions? This includes indentation, variable naming, variable typing, method naming, and other things I’m probably not thinking about currently..