Dukosi Coding Challenge - Minesweeper

# Minesweeper Overview and Rules

Minesweeper is a graphical game where the player must hunt and identify all the mines on a grid made up of squares.

The computer allocates a fixed number of mines to squares within the grid at the start of a game, with each square containing either 1 mine or 0 mines. Whether a square contains a mine is hidden from the player.

The player is told how many mines are on the grid. The player’s objective is to clear all squares on the grid that do not contain a mine (safe squares), leaving only those squares that contain a mine (unsafe squares).

If the player attempts to clear a square containing a mine, then the player loses the game and the game is over. If there was no mine at that square then the total number of mines in each adjacent square is added and displayed in that square.

The adjacent squares are the 8 squares found immediately to the left, right, above, and below a location. For square X in Figure 1, the 8 adjacent squares are highlighted in blue.

A picture containing shoji, crossword puzzle, building

Description automatically generated

**Figure 1: Adjacent squares (blue) surrounding square X**

The player may flag a square in which they believe there is a mine. This is for the player’s own information only and it has no effect on whether the game is won or lost.

The player wins the game when all safe squares are cleared, leaving only unsafe squares.

## Example Game Outcomes

Figure 2 shows an example of a game that has been lost. The player cleared a square containing a mine ( ).

Graphical user interface, application, Teams

Description automatically generated

**Figure 2: Example of a losing state**

Figure 3 shows an example of a game that has been won. The player cleared all safe squares, with only unsafe squares remaining.

A screenshot of a phone

Description automatically generated with low confidence

**Figure 3: Example of a winning state**

# The Task

## Game Module

A module should be written in to manage the state of a game of minesweeper. This module should:

* Allow a new game to be started, setting up the initial state of the game
* Expose functions that allow a player to take the actions permitted by the rules of the game and update the game state accordingly
* Expose the game state in a way that would allow a GUI application to use this module to draw the game (note that a GUI should not be implemented)
* Indicate whether the game is won or lost
* Include suitable data items and data structures to achieve the above

Assumptions are provided below (if you find you need to make additional assumptions, then document them as comments within the code):

* The game uses a grid that has a fixed size of 8x8 squares
* There are 10 mines hidden on the grid
* The locations of mines within the grid may be input to the module when a new game is started (your module does not need to randomly allocate mines within the grid)
* When a safe square is cleared, most implementations of minesweeper recursively clear any other safe squares adjacent to this - your implementation does not need to do this.

## Test Application

An application should also be written to test your game module.

Write sufficient tests to thoroughly test the game module. Your test application should work through the test cases and report the test results.

## Development Environment

If you are applying for a firmware development role, then you should write your code in C. If you are applying for a Software Tools or Software Test Engineer role, then you should write your code in Python. Use a development environment and toolchain of your choice. Include short instructions that specify how the environment/toolchain is used and how to build, as well as the expected output when the application is run.

Use of the standard C libraries is allowed, and this will be useful for outputting test results.

## Evaluation Criteria

The code and other files that make up your submission should be your best work.

It is up to you how long you wish to spend on the task before submission. We are all busy, so it is acceptable to leave out features or pieces of functionality provided that their absence is documented. Your submission will be evaluated on the parts that are present.

Criteria:

1. Is the code functionally correct?
2. Is the code easy to follow and maintain?
3. Is the module easy to use (if someone wanted to make a minesweeper GUI, would it be easy for them to integrate your module)?
4. Is the implementation efficient in use of resources (compute time and memory)?
5. Is the implementation portable?
6. Is the implementation secure?
7. Is the code commented well (architecture and design should be clear from comments within the code)?
8. Is the code consistent?
9. Is the code functionally complete?

## Submission

Store your code, other files required to build your application, and any accompanying documentation in a zip file.

Please email this zip file to [francisbain@dukosi.com](mailto:francisbain@dukosi.com)

Good luck and happy coding :)