TriTech Software Interview Exercise: Minesweeper

Welcome and Overview

Welcome to the TriTech Software Minesweeper Exercise. Please read the following notes and instructions carefully before beginning. If you have any questions, please Reply-All to this message and we'll get back to you with an answer as soon as possible.

This exercise is designed to help you showcase your front-end software development skills. Consequently, we leave the implementation largely open-ended: you may use any languages and tools you are familiar with, as long as your solution is browser-based. We also respect your time, and do not expect you to spend more than around 3 hours on this exercise.

For completeness, all of the game rules and features for Minesweeper are described. We do not anticipate or expect that you will implement all features of the game. The overall quality of your solution is much more important.

Once you have submitted your solution, we will schedule a call over Microsoft Teams to review your implementation. Please be sure you have a working microphone and camera and are able to share your screen. Be prepared to discuss design decisions, specific coding techniques, and other aspects of your solution.

Provide your solution via a git repository. This can be a hosted repo on GitHub, GitLab, etc. or a local repo.

If your solution is hosted, once you have completed the exercise send an email with a link to your repository to:

twheeler@tritechsoft.com, scrim@tritechsoft.com, hchurch@tritechsoft.com

If your solution is a local repo, create an archive of it that includes the `.git/` directory. Please DO NOT include the `node_modules/` directory if it is in your solution! Then go to https://filepost.io/, upload the archive, and send the generated link to:

twheeler@tritechsoft.com, scrim@tritechsoft.com, hchurch@tritechsoft.com

Solution Requirements and Expectations

- Your solution must be browser-based, i.e., will work in a modern web browser like Chrome, Firefox or Edge.
- You may use any languages and tools you are familiar with to implement your solution.
- Provide your solution via a git repository. See instructions above for submission.
- Use commits to show your work. Each commit should be a small, logical unit of work.
- It is *not* expected that you will implement all features of the game. However, your solution should be runnable.

- Develop your solution using professional standards. Consider the structure and quality of your code. Is it readable and maintainable? Does it follow good practices?
- There is no need to create a deployment pipeline, but do take into consideration production-readiness and the effort needed to drop this into a pipeline.
- Be prepared to discuss design decisions, specific coding techniques, and other aspects of your solution during the review session.

Minesweeper Game Rules

Below are the complete rules and features of the game. You are not expected to implement the entire game.

The game of Minesweeper is played on a grid of 10 rows and 10 columns. Ten squares contain a bomb. The player uses logical analysis (and, sometimes, guesswork) to determine which squares contain a bomb. The goal of the game is to correctly identify and 'flag' all squares that contain a bomb, and reveal all other squares, without revealing a square that contains a bomb.

Each square in the grid is initially marked as 'closed' and its contents are unknown to the player. The player uses the left mouse button to 'open' a square and the right mouse button to 'flag' a square. Opening a square reveals its actual contents, as follows:

- If the square contains a bomb, it explodes and the player has lost the game.
- Otherwise, the square is opened and shows the number of adjacent squares (including diagonal) that contain a bomb. If no adjacent squares contain a bomb, the square is left blank and additional adjacent squares are automatically revealed in the same fashion as if they were clicked. If any of these automatically-revealed squares also has no adjacent squares with a bomb, they are also automatically revealed.

Flagging a square by right-clicking it identifies it as containing a bomb. The game does not indicate whether a flagged square actually contains a bomb, and it is possible to flag a square that does not have a bomb.

If the player correctly 'flags' all squares that contain a bomb, they win the game. If they 'open' a square containing a bomb, it blows up and they have lost the game.

If the player wins, their elapsed time determines their score, with shorter times being better.

Before continuing, visit https://minesweeperonline.com/#beginner to familiarize yourself with the gameplay.

Detailed Feature Description

The following detailed list of features and behaviors may be helpful in design and implementation of your solution.

- The game is played on a rectangular grid of 10 x 10 squares. Ten randomly selected squares contain a bomb.
- Each square is initially *closed*. Its contents are not visible to the player.

- The game timer begins the first time the player either *opens* or *flags* a square.
- Typically, a square is opened by left-clicking the mouse and a square is flagged or unflagged by right-clicking the mouse. However, this is not required.
- When a square is flagged, an icon is placed on the square to visually indicate that the player believes it to contain a mine.
- A flagged square cannot be left-clicked to reveal its contents.
- A flagged square can be marked as closed by flagging it again.
- When a square is opened, its contents are revealed:
    - If the square contains a bomb, it explodes and the player has lost the game.
    - If any of the opened square’s neighbors, including diagonally, contains a bomb, the opened square indicates the number of adjacent bombs.
    - If none of the opened square’s neighbors, including diagonally, contains a bomb, the opened square indicates blank. The opened square’s neighbors are then revealed and the same procedure applies to them.
- The player wins the game when the following conditions are met:
    - All squares have been opened or correctly flagged.
    - All squares that have been flagged contain a bomb.
- The player loses the game when they open a closed square that contains a mine.
- When the game is over:
    - The game timer stops.
    - All remaining closed squares are opened to reveal their contents:
        - A square that does not contain a mine is revealed normally as if the player had opened it.
        - A square containing a mine displays a mine in it.
    - All squares that were correctly flagged as mines are unchanged.
    - All squares that were incorrectly flagged as mines are revealed as mines with a red ‘X’ overlayed.

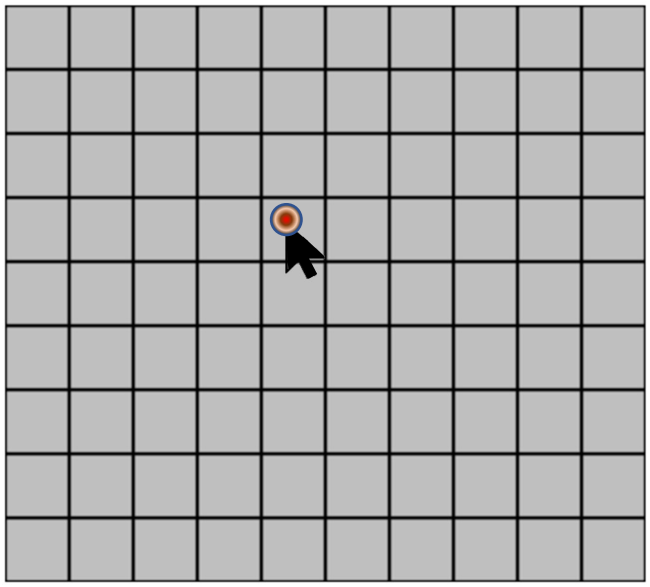
Icons

You may use these assets in your solution:

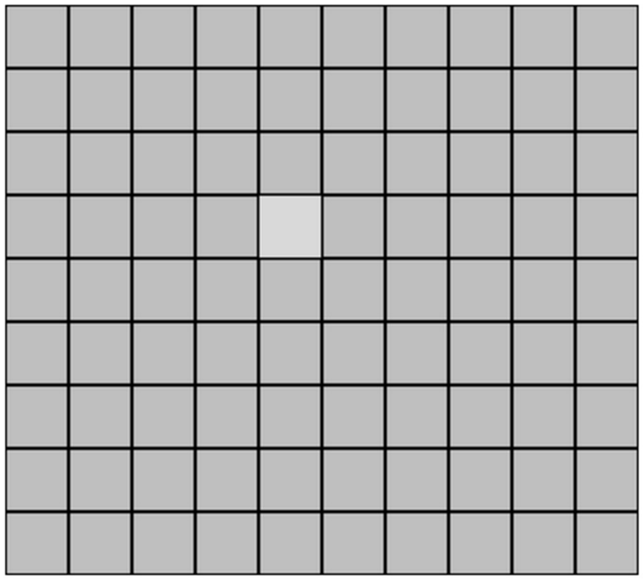
Revealed Bomb	Incorrectly Marked Bomb	Exploded Bomb	Flagged Bomb
💣	❌	💥	🚩
Closed	**Open**		
■	□		
One	**Two**	**Three**	**Four**
1	2	3	4
Five	**Six**	**Seven**	**Eight**
5	6	7	8

Revealing a Square

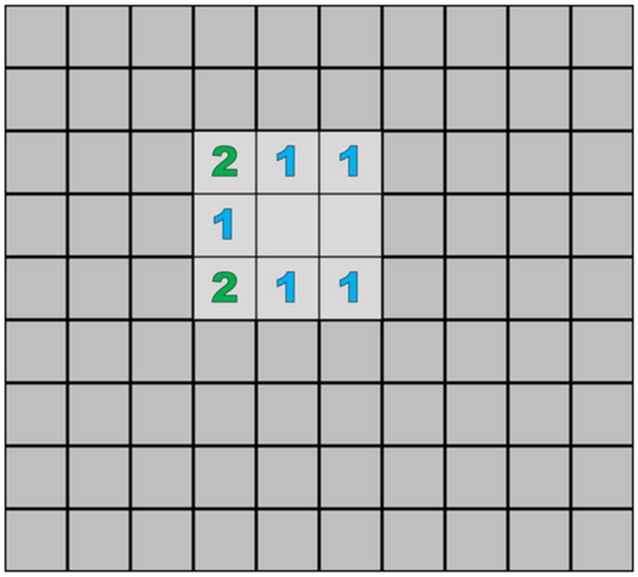
Beginning with all squares closed, the player selects left-clicks to open the square at (4, 5):



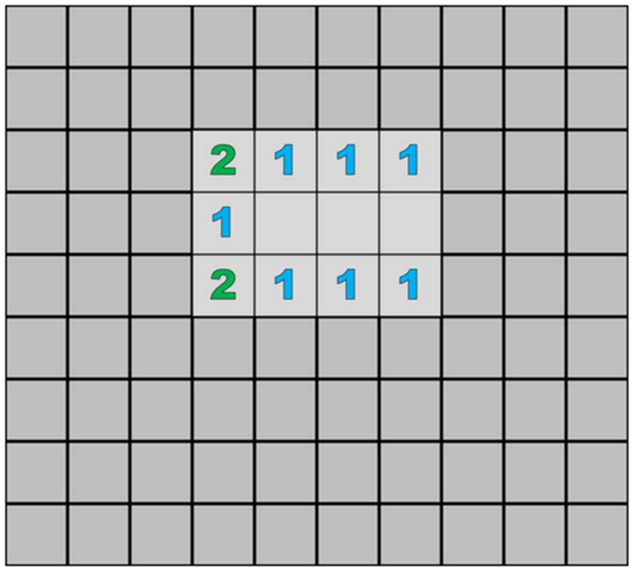
The square is opened, and there are no adjacent squares containing a bomb:



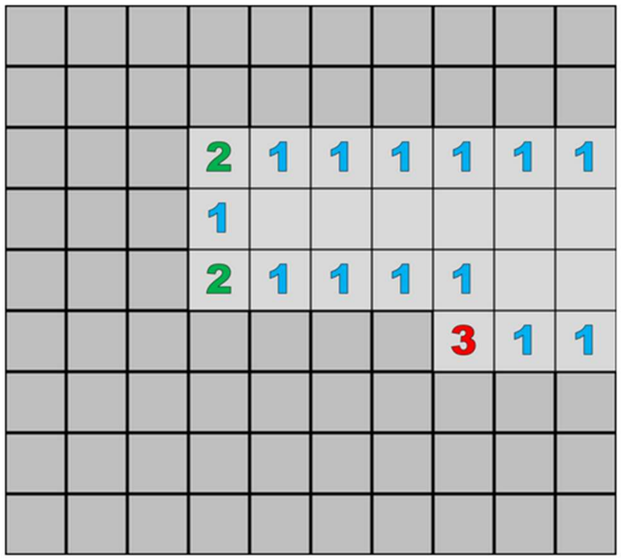
Since there are no adjacent squares with a bomb, the squares adjacent to (4, 5) are automatically revealed. Note that square (4, 6) also has no adjacent bombs:



Since (4, 6) has no adjacent bombs, its adjacent squares are also automatically revealed:



The process of automatically revealing squares continues until all the automatically-revealed squares contain a number:


2 1 1 1 1 1 1
1
2 1 1 1 1
3 1 1