# The Test

**Set aside 2 hours to create some code that shows how you would code a minefield/minesweeper style game running on the command line (no UI), in order to demonstrate how you would code & test a real-world application using established best practices**.

In the game a player navigates from one side of a chessboard grid to the other whilst trying to avoid hidden mines. The player has a number of lives, losing one each time a mine is hit, and the final score is the number of moves taken in order to reach the other side of the board. The command line / console interface should be simple, allowing the player to input move direction (up, down, left, right) and the game to show the resulting position (e.g. C2 in chess board terminology) along with number of lives left and number of moves taken.

**Above all else please follow these guidelines**

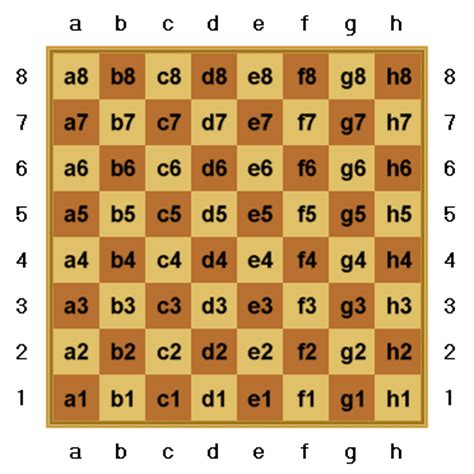
1. **Quality is more important than quantity**
2. **We will assess your ability to write clean-code that has good structure & is covered by meaningful tests**
3. **Don’t code a UI**

When complete, upload your code to a public GitHub repository and forward the URL to us.

Be prepared to talk through you code and explain key design features and coding principles and why you have used them.

Good luck!

Notes

1. A chess board looks like this:   
   
   1. 8x8 (64) squares on a board
   2. X axis (columns) labelled with letters A to H, left to right
   3. Y axis (rows) labelled with numbers 1 to 8, bottom to top
   4. Each square is labelled by combining the X axis letter with the Y axis number e.g. A1 is the bottom left square.
   5. At the start of a new game the board should be built with 64 new squares (Square factory + config).
   6. Squares can have different features:
      1. A square with a mine will result in different outcomes should a player enter it:
         1. If the player has more than one life left. One life will be lost.
         2. If the player has no more lives left, the game is over.
      2. A square will support a player entering it:
         1. The players score will increment.
         2. The player will have a new position on the board.
         3. The player will have a new set of navigation options depending on the square’s location.
         4. If the square is in the final row e.g. A8, the player will have won. The game will be over, the score reported and maybe added to the leader board :-D. Nb. Squares in the final row won’t have the capability to facilitate the player navigating.
      3. A square will support the ability for the player to navigate across the board according to possible moves.
   7. A player starts on a random square in row 1 e.g.