Exam front end developer Angular/ReactJS

You have 3-4 hours to finish the test.

In this test you will be required to solve various problems utilizing Angular/React. **Using third party libraries are not allowed** unless explicitly required by the test!
Please make sure to spend your time wisely on each question (answer bonus sections only if you have finished questions fully), in case you are stuck it is recommended to move on to the next question.

Please remember that even if you don't have any prior experience with the following concepts try to solve them in the best possible manner.
Remember that it is more important for us to see what you have accomplished and the approach you have taken for finding the solution.

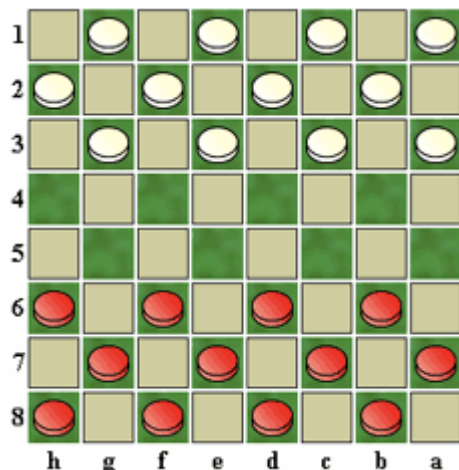Please solve the following questions:

Checkers Game   .1
1.1 Create a checkers game board dynamically. Use the following image as a reference:
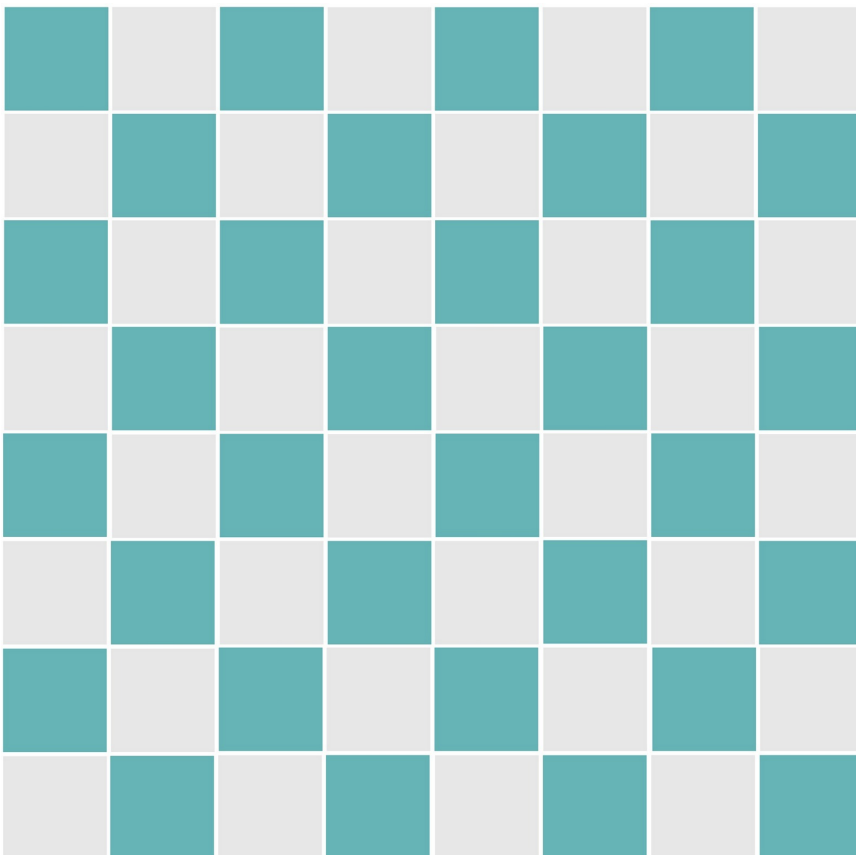
1.2 Create dynamic "checkers" for 2 players - according to checkers game rules (use the following description for checkers game rules)

**Setup and Object of Checkers**

Checkers is played on a standard 64 square board. Only the 32 dark colored squares are used in play. Each player begins the game with 12 pieces, or checkers, placed in the three rows closest to him or her.

The object of the game is to capture all of your opponent's checkers or position your pieces so that your opponent has no available moves.

*You are <u>not</u> required to create 3d players - they can be simple circles.

1.3 Allow the "players" to move according to checkers game rules on the board (use the following description for checkers game movement rules)

Movement
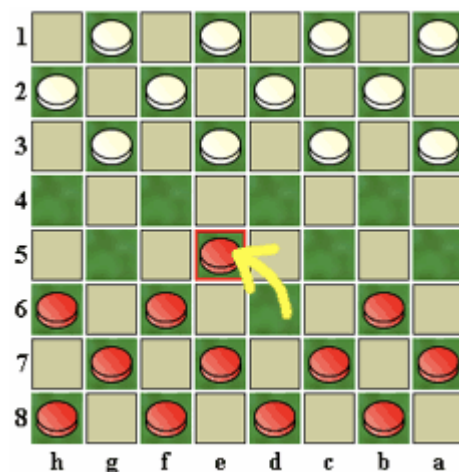
Basic movement is to move a checker one space diagonally forward. You can not move a checker backwards. If a jump is available, you must take the jump, as described in the next question and answer.

1.4 Allow the "players" to jump over the opponent's checkers/players in order to capture/eat them
according to checkers game rules (use the following description for checkers game jumping rules)

Jumping
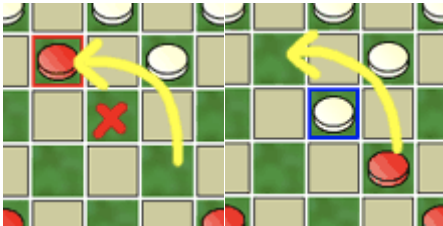
If one of your opponent's checkers is on a forward diagonal next to one of your checkers, and the
next space beyond the opponent's checker is empty, then your checker must jump the opponent's
checker and land in the space beyond. Your opponent's checker is captured and removed from the
board.

*Please note that you do not need to support multiple jumps in one move like the following:
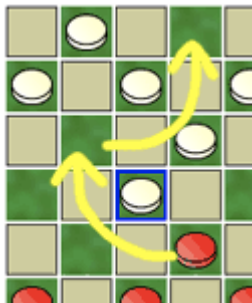
***Do not implement any extra game rules that are not written explicitly in the document, It is not
required!**

Redo/ Undo - allow game state to be undone or redone.   .2

2.1 Add a "Undo" button - each press will change the game state one "move" backwards (if possible).

2.2 Add a "Redo" button - each press will change the game state one "move" forwards (if possible).

2.3 Add a "Start Game" button - each press starts/restarts the board state and resets the game.

2.3 Bonus:

Animate each "players" move.   -
Animate removal of "players" from the board.   -
On each press on undo/redo use "player" animation while changing the game state.   -