CMSC 15100 Introduction to Computer Science I
Homework 9

You should submit a single .rkt file with your code via Canvas. The homework may require more debugging than usual, so it's recommended that you start sooner rather than later. At the top of the file for this assignment include a comment,

```
;; Homework 9
;;
;; your name
;; your CNET id
;; your collaborators
```

If your code doesn't compile, it may not be graded. Please double check to make sure your code runs before submitting!

Tasks

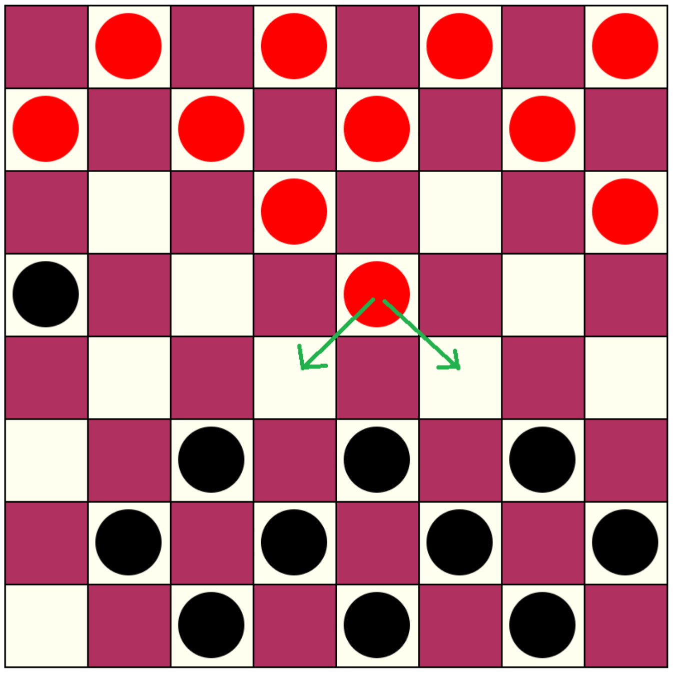
For this assignment, you will finish implementing checkers. Starter code for this homework has been posted on Canvas, but you are encouraged to pick up from your previous submission for Homework 7. If you didn't do the extra credit, you may want to replace your `draw-board` function with the starter code's `draw-board` function so that you have the extra visuals.

You are encouraged to write helper functions as necessary to complete these tasks. As before, there's no need to submit tests for any of this assignment. To make sure your code works, run `(start-game)` and test your code visually.

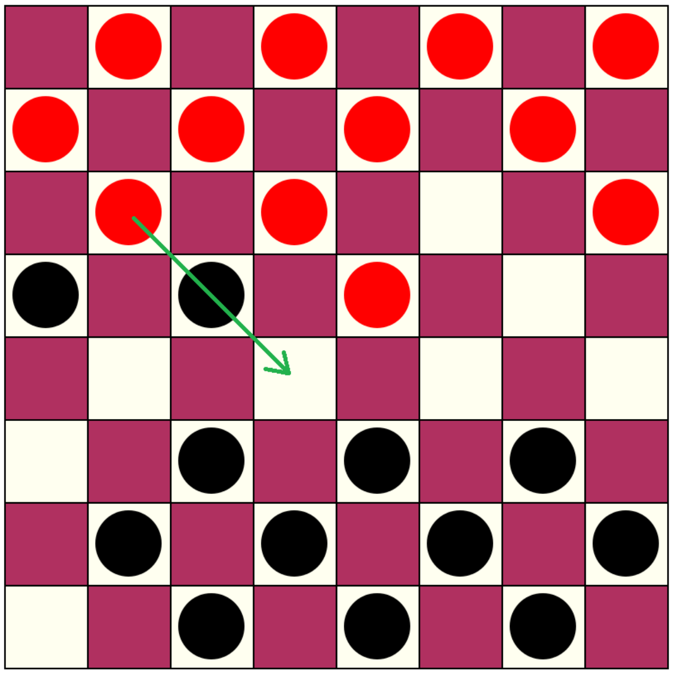
1. (22 points) The most glaring omission from the previous assignment on checkers is that moves could be made anywhere! Amend this.

   (a) Implement a function `valid-move?` which takes two inputs, a `Checkers` game and a `Loc`, and returns whether or the `clicked-piece` can move to the given `Loc`.

   In checkers, there are two types of valid moves, which we call moves and jumps. In a move, a piece moves diagonally towards the opponent's side of the board into an unoccupied square. For example, the green arrows here are valid moves,



In a jump, a piece again moves diagonally towards the opponent's side of the board, but it jumps over one of the opponent's pieces. Note that you **cannot** jump over your own pieces. In this picture, a green arrow shows a valid jump by a red piece,



Hint: you may want to use the `get-piece-or-none` function as a helper function

(b) Modify the `click-board` function to check if a move is valid before calling `place-piece`. If the move is invalid, the game state should be returned unmodified.

   (c) Modify `place-piece` so that, if the piece is placed two squares diagonally ahead, any intervening piece is removed from the game.

2. (7 points) Next, we need to make the moves alternate between players. This can be accomplished by checking and updating the `turn` field inside the `Checkers` object.

   (a) Change the function `valid-move?` so that only the player whose turn it currently is can make a move.

   (b) Change the function `click-board` so that only the player whose turn it currently is can click a piece.

   (c) Change the function `click-board` so that, if a piece is played, the `turn` field changes from `'red` to `'black` or vice versa.

3. (6 points) Finally, if one player has no pieces remaining, the game has ended and we would like our window to exit and close. Implement a function `game-over?` which takes a single input of type `Checkers` and returns true if the game is over.

   In the starter code for this assignment, this function will be automatically called by universe.rkt. If you are using your own code from Homework 7 instead of the starter code, change the `start-game` function at the very bottom of the file from

```
(define (start-game)
  (big-bang starting-board : Checkers
    [to-draw draw-board]
    [on-mouse click-board]))
```

   to

```
(define (start-game)
  (big-bang starting-board : Checkers
    [to-draw draw-board]
    [on-mouse click-board]
    [stop-when game-over?]))
```