Assignment 6
Due: Sunday, 18 March 2018, 11:59 PM on TEACH as a .cpp and README.txt
Checkers

Set Up

- Two players, one has red tokens, the other has white tokens.
- Checkers may be played on an 8x8, 10x10 or 12x12 board.
- No matter the size of the board, pieces are placed on the dark squares of the board, leaving the middle two rows blank.

How to Play (Recommended: https://www.youtube.com/watch?v=m0drB0cx8pQ)

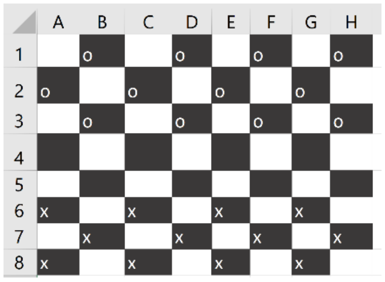
- Players cannot move an opponent's piece.
- Turns alternate between players until the game is won.
- The player who runs out of pieces first **or** cannot make a valid move loses the game.
- Pieces may only move diagonally into unoccupied squares and only by one square if they are not capturing a piece.
- A piece may capture opponent's pieces by jumping over them diagonally. So long as there are successive pieces to be captured, the piece may continue moving. The line the piece travels does not have to be straight along one diagonal, it may change directions so long as it continues to progress the board and capture pieces.
- When a piece has reached the King's Row (furthest row from it's originating side) then it is "kinged" by placing a second token on top of it. This piece is now permitted to move backwards on the board and allowed to capture backward.

Implementation

- Establish the size of the board via command line arguments, must include error handling for too many and too few arguments as well as incorrect input.
- The board must be created using two-dimensional, dynamic memory.
- The board must be correctly colored black and white using the following code as a base. The following code fragment colors a two-dimensional board. It is expected that you will adjust it as needed to provide the best user interface possible for your program.

```
for(int i=0; i<rows; i++){
     for(int j=0; j<cols; j++) {
          if (i%2 == 0 && j%2 == 0)
               cout << "|\033[30;47m " << board[i][j] << " ";
          else if (i%2 == 1 && j%2 == 1)
               cout << "|\033[30;47m " << board[i][j] << " ";
          else
               cout << "|\033[0m " << board[i][j] << " ";
          cout << "\033[0m";
     }
     cout << endl;
}
```

- The game must be correctly set up and played as described in the “Set Up” and “How to Play” sections of this document.
- The path of travel shall be received as a single, c-style string and correctly parsed and error handled to move the pieces in a correct fashion on the board. The string should be a list of square coordinates. It is up to you as the programmer to determine the best delimiter and coordinate system to use. Whatever method that is chosen should be clearly communicated to the user. The board should be printed after each move of a piece and include the defined coordinate system. This is especially important for a piece that may move multiple squares in a turn. After each move, captured pieces should be removed from the board. Example board below:



	A	B	C	D	E	F	G	H
1		o		o		o		o
2	o		o		o		o	
3		o		o		o		o
4								
5								
6	x		x		x		x	
7		x		x		x		x
8	x		x		x		x	

- When the move sequence of a turn has been completed, the total number of captured pieces for each player should be displayed.
- If a piece makes the King’s Row then it should be “kinged” in some way. Kinged pieces are the only pieces which may move forwards and backwards on the board.
- The game should end when only one color of piece remains on the board or if a player has run out of valid moves.
- You will submit a README.txt that outlines how your program is compiled and run.

Grading

Design to be submitted to recitation.

Implementation – 80 pts

Automatic Deductions

Memory Leaks: -10 pts

Segmentation Faults: -10 pts

Global Variables (including constants): -10 pts

Missing Error Handling: -10 pts

Functions More Than 20 Lines: -10 pts

Missing README File: -10 pts

Style and Code Quality – 20 pts

Assignment 6 will not be demoed. TAs will grade on their own during Finals Week. You will submit a README.txt that outlines how your program is compiled and run. Failure to submit a README.txt will result in a deduction as well as any penalties that may be incurred as a result of incorrect use of your program.