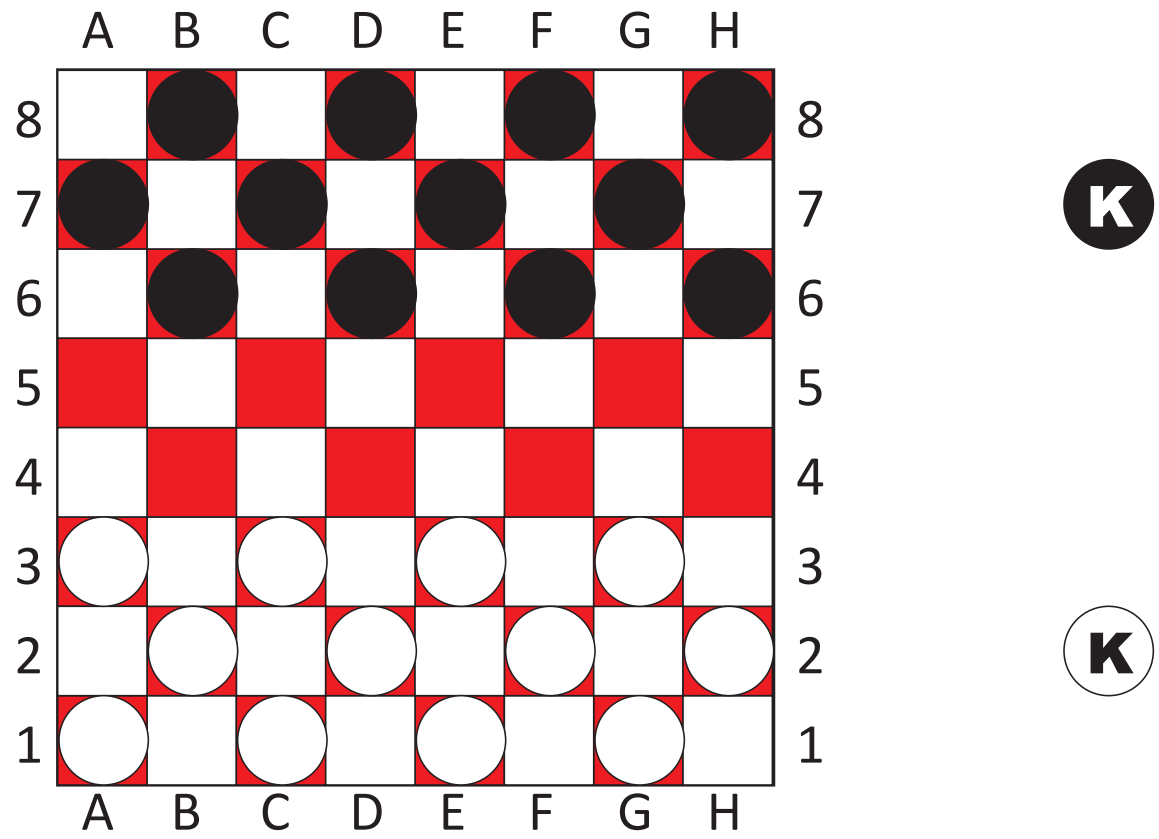
# CS 2ME3 and SE 2AA4: Assignment 3 January – April 2014

**Due: 9 April, 2014**

Assignment 3 builds on the work you did in Assignments 1 and 2.




**Repeated from Assignment 1**

6. In order to help you with the decomposition, here is some indication of what is to come in assignments 2 and 3:
   6.1. Assignment 2 will require you to be able to move pieces on the board. The moves have to be legal moves. You can move pieces using a graphical interface, or by using accepted checkers terminology, i.e. E3-D4 (move the piece on E3 to its new position, D4).
   6.2. Assignment 3 will require you to provide two modes of operation: 2 player checkers where 2 people can play against each other; or 1 player against the computer. The latter mode will require you to include an automated mode in which algorithms are used to determine the moves for the computer.

**Specific requirements for Assignment 3**

1. Include choices that enable users to:
   1.1. Start a game from original starting positions.
   1.2. Start a game from a previously stored state (the state should be saved within a file – and you can decide whether you want to allow more than 1 saved game).
   1.3. Two people play against each other (in other words, the user interface must allow the players to input their moves, in turn).
   1.4. One person plays against the computer. Let the user choose which colour pieces they want to use. The player with the white pieces moves first.
   1.5. Make moves – the moves must be legal moves. You should be able to make moves that: simply move a piece to another square; jump the opponent's piece (so that piece is removed from the board); convert a piece to a "king"; move kings in both directions (forwards and backwards). You can decide how you want the user to indicate moves – graphically or by code (E3-D4 etc), or both. Note that the rules of checkers are such that if a capture move is possible it must be taken. If more than one such move is possible, the player chooses which one to make.
   1.6. Save a game to be resumed later.
   1.7. Indicate if the game has been won. Note that a legal move is to resign.
   1.8. Include any on screen help you think is necessary/useful.
2. The deliverables for the assignment include:
   2.1. Extensions to your existing design documents. Make sure you find a way to tell the reader (and yourselves) what has changed in your design. A revision history is a good idea.
   2.2. Extensions to your existing code. Make sure you find a way to tell the reader (and yourselves) what has changed in your design. A revision history is a good idea.
   2.3. Include a test report document that records how you tested your application. By the time you complete this assignment you will know much more about testing. Make sure you explain the coverage of your test cases.

**Grading**

The assignment will be graded out of 50.

30 marks for the design document (description of changes from assignment 2 – 5, decomposition – 5, public interface – 4, uses relationship – 2, private implementation – 4, traceability – 2, evaluation of the adequacy of the design – 8).
5 marks for the code – general quality.
15 marks for testing.

*Bonus:* 5 marks for a "reasonable" algorithm for computer to make moves.
2 marks for including time limits.
(Note: max grade for the assignments is 20% of the total course grade.)

**Special notes on documentation**

- A module guide, MIS and MID for each class is the minimum documentation for the design. Uses relationship, traceability matrix, traceability of secrets are goals to aim for.
- Document the reference you used for the rules of the game.