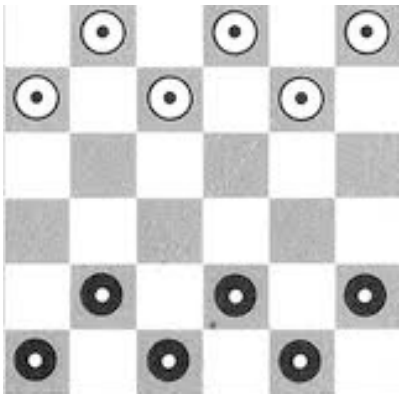
CS 4613 * CS 6613 **Spring 2018**
Project: Mini-Checkers Game **E. K. Wong**

Total points: 100. Optional Extra credits: up to 30 points.

Project Description: Design and implement a *Mini-Checkers* game for a human to play against a computer. The game is played on a reduced game board size of 6 x 6 squares and the rules have been modified from the original 8 x 8 checkers game.

- The checker board consists of 6 x 6 alternating light and dark squares. The board is placed so that the left corner square on each player's side is a dark square.



- Each player starts out with six playing pieces, with the white pieces for the computer and the black pieces for the human, arranged as in the figure above. Note that all the pieces are placed on dark squares.
- At the start of the game, the human player can choose to move first or second.
- Each player takes turn to make a move. There are two types of moves: *regular moves* and *capture moves*.
    - In a regular move, a piece can move forward diagonally to an adjacent square that is empty.
    - In a capture move, a piece can jump over and capture an opponent's piece and land on an empty square (landing on a square that is not empty is not allowed.) The jump must be in the forward diagonal direction and no consecutive jumps are allowed. In addition, every opportunity to jump *must be taken*. In the case where there are two or more possible jump moves, the player can choose which one to take.
    - No vertical, horizontal or backward moves are allowed for both regular and capture moves.
- If a player has no legal move to take, his/her turn will be forfeited and the other player will make the next move.
- A player wins when he/she captures all of the other player's pieces. If both players do not have any legal move to take, the game will end and the player with the most number of pieces left wins; if the two players have the same number of pieces left, the game is a draw.

(In the original 8 x 8 version of the game, once a piece reaches the far end of the board, it becomes a king. We are not implementing this feature in the project.)

Implementation: Implement the *Alpha-Beta Search Algorithm* we covered in lecture to determine the *next best move* for the computer. Every time your program calls the *Alpha-Beta Search Function* to determine the next best move, outputs the following statistics for the game

tree generated: (1) maximum depth of tree (let the root node be level 0), (2) total number of nodes generated (including root node) in the tree, and (3) number of times pruning occurred in the MAX-VALUE function and (4) number of times pruning occurred in the MIN-VALUE function. At the end of the game, the program should declare the winner or a draw (in case of a draw.) As a minimum requirement, your program will run in *command line* mode that will allow the human player to choose a piece and specify the board position to move the piece to. C++ , Python, or Java are the recommended programming languages. If you plan to use another language, send me an e-mail first.

If your program cannot finish computing for a move within **15 seconds**, then you should cutoff the search at certain depth level *l* and use an evaluation function to estimate the expected utility value for nodes at the cutoff level. You are to **design the heuristics** for use in the evaluation function. You should set the cutoff level *l* as deep as possible, yet allowing your program to finish computing within 15 seconds. If you use cutoff, your program should output (1) the cutoff level and statistics (2) to (4) as described above.

Extra Credits: You can implement one or both of the following to get extra credits. This is *optional.*

1. [15 points] Design and implement a graphical user interface (GUI) that displays the 6 x 6 game board and the checkers pieces, and use the mouse to pick and move the pieces.
2. [15 points] In the design above, the computer plays the best strategy. Design your program with three levels of difficulty, from 1 to 3, so that it is easy for the human player to win at level 1 and difficult for the human player to win at level 3. The human player can choose the level of difficulty at the beginning of the game.

Hand In:

1. Source code for your program with in-line comments. Points will be taken off if you do not have in-line comments.
2. An MS Words or PDF file that contains:
   - Instructions on how to compile and run your program.
   - A high level description of your design and your program.
   - Definition of your terminal states and their utility values.
   - If you use an evaluation function, explain how your evaluation function works and the heuristics you use.
   - If you implement different levels of difficulty, explain how they are implemented.

Computer Demo: A 15-minute demo of your program running on your computer will be required. Instruction to be announced later.

You are to work on this project by yourself. You can discuss with your classmates on how to do the project but everyone is expected to write her/his own program and report.