Project: Mini-Checkers Game

High Level Design and Description:

Mini-Checkers Game is for human to play against computer. It has 6*6 game board with alternative light and dark squares. On dark squares pieces are being placed. The arrangement of light and dark square is made such that left end of each player will have piece on it.

Piece can move only in diagonal-forward direction. Each piece can take jump or normal move. Normal move is consisting of diagonally moving one step forward. Piece can take jump if opponent is present in next diagonal position and next to next position is empty. For any piece there are four possible moves jump-left-forward, jump-right-forward, move-left-forward, move-right-forward. If left or right jump is possible then code doesn't check for normal moves. If jump not possible then code records normal moves. Among all possible moves code checks for possible legal moves by checking board size and position being empty to take the move.

Black piece represents human player and white piece is for computer player. Human player and computer play alternatively. In case of any one goes out of moves, chance will be given to other player. When both player goes out of move winner will be decided based on no of pieces remaining. Whoever has more no of pieces left on the board wins the game. If same no of pieces left for both the players, result of the game will be draw.

Human player can select whether to play first or second. Human player selects his move on GUI by selecting which piece to move and where to move. All the movable pieces will be highlighted on the board to make it easier for human to take move. Computer player takes move by following alpha beta search algorithm. This algorithm returns best action to be taken and Computer player takes that action. The algorithm builds entire tree and gives output of total nodes generated and no of times pruning took place. After each player plays its turn the GUI is repainted.

Run Instructions:

On your command prompt go to the path where you have kept the source code using **"cd "** instruction.

To compile the code, write below instruction on your command window:
"javac CheckersGame.java"
This will create CheckersGame.class file.

To run the code, write below instruction
"Appletviewer CheckersGame.html"

This will open the applet window and you can select to play first or second. If you select play first option you will get a window with your legal moves highlighted. If you select second computer will take its move and you will get chance with your highlighted legal moves.

Terminal State and Utility Value:

During alpha beta search algorithm if I get legal action for any state as null then that state is terminal state. If terminal state is found I am finding utility value of that state. Since Utility value is found for computer player I am taking utility value as (no of computer pieces remaining – no of human pieces remaining). If human pieces are 2 and computer pieces are 3 then utility function returns 3-2= 1. Which gives positive value if computer is winning and negative value if computer is losing and in case of draw it return 0.

Since my alpha-beta algorithm gives me result within 15 s, I haven't used cutoff and evaluation functions. Thus there are no heuristics used.