Assignment #1

Due date: 10/8

Total Score: 100 + (20 bonus points)

//

Objectives: Implementing a **mini-max algorithm** and **heuristic functions**

Problem: Write a program to play a relatively simple chess endgame. *Player X* has a rook and king and *Player* Y has only the king left. Of course, with more chess pieces, Player X will likely win this game. So Player X tries to win the game as quickly as possible avoiding the infinite loop or dead end. On the other hand, Player Y tries to play the game as long as possible by delaying it. This assignment can be completed individually or by a team with max. 5 members.

Requirements

1. Your program should define at least functions, moveWhite, moveBlack, play, heuristic, heuristic, and printBoard. Functions, **moveWhite** and **moveBlack** should have parameters for the positions of the chess pieces and return the modified chessboard after moving the necessary chess piece(s) based on the parameters. Function, **play (*n*)** is a driver function that starts and controls the game by alternating moveWhite and moveBlack functions up to ***n*** number of moves, where n is a parameter for the max. number of moves allowed for each player in this game. The function play should immediately stop the game either when "checkmate" or "stalemate" occurs even before the *n*th move or after a player makes *n* number of moves. When the king is under attack and there is no applicable move, it returns the "checkmate". In this case, player X wins the game. When the king is not under attack and there is no applicable move except for the move that would put or leave the king under attack, it returns the atom "stalemate". In this case, the game ends in a draw.

   Define two heuristic functions, **heuristicX** for player X and **heuristicY** for player Y, each of which returns a heuristic value indicating the usefulness of a move based on the current chessboard. In addition, define a function **printBoard** that prints the state of each move during the game in a nice 8x8 board form to a file and the screen.

   Properly modularize the program clearly separating it into different modules (classes, package, or dll) for related data structures for the main algorithm; move functions; heuristic functions; reporting function so that the program can be easily modified or maintained in the future. In addition, document the program properly naming and commenting each module or function so that one can easily understand the intention.

2. Test your program for the following 4 test cases with ***n*= 35** for two possible scenarios specified in (a) and (b)**:**

   Test case1: w(5,6), r(8,5), b(6,8), Test case2: w(6,5), r(5,6), b(4,7), Test case3: w(7,6), r(8,5), b(7,8)
   Note: Player X has the first two pieces and Player Y has the last piece.

   (a) Only **heuristicX** is used. If the king is under attack, move the king to any safe position to avoid immediate "checkmate". Otherwise, the king moves randomly.
   (b) Both **heuristicX** and **heuristicY** are used. In this case, Player X tries to win the game as quickly as possible. Player Y tries to delay the game as late as possible.

The reporting function should display the trace of running program in a nice format printing at least the test case, name of heuristic function used, number of moves made, the game result, and each state of the chessboard as shown below:

~~~~~~
Game started...
Testcase1: w(5,6), r(8,5), b(6,8)
Heuristic function used: heuristicX
~~~~~~
Board => Board => Board => Board => Board => Board => Board => Board => Board

Number of moves made: 10
Game result: checkmate

Note: Board represents an actual chessboard with chess pieces placed on it.

3. Write a brief report in **Word format** including (a) the name(s) and contact email addresses of the developer(s); the percentage contribution to this assignment if the assignment was completed by a team. If a team cannot reach a consensus on the individual contribution, include the individual's claimed percent contribution with a brief description on specific tasks performed (b) how to run the program (c) discussion on the algorithm, analysis for each heuristic function specifying the main strategy and its effectiveness, (d) reference to the source of the program if some portion or all of the program was reused (or copied) from other people's code specifying the URL, author if any, (e) optionally comments or lessons learned from this assignment.

   Warning: Although the code reuse is allowed for this assignment, copying the code from other person or team in this class is strictly prohibited. Any one or team violating this rule will receive ZERO score from this assignment.

   Note: (a) ~ (c) are required and (d) ~ (e) are optional.

   Bonus points will be given to top 3 programs that won all 3 games by the smallest number of moves under scenario (a).

//How to Submit this Assignment//
Zip a report, source code, and an executable program into **ONE file** and submit the zipped file **to Titanium** by the due date. I strongly recommend you to write the report in **Word format** so that I can provide my feedback directly in the report. If a PDF format of report is submitted, no feedback will be provided.

Grade will be based on the quality of the program, heuristic functions, and report.