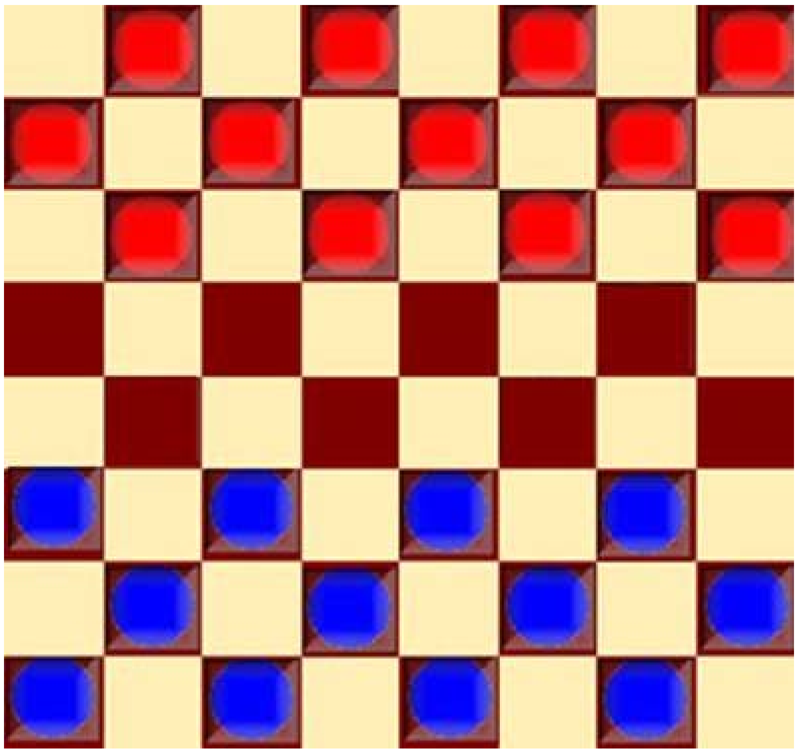
CSC 680: Artificial Intelligence
Fall 2012
Assignment #3 (30 Points)
Due date: Wednesday October 17, 2012 (in class)

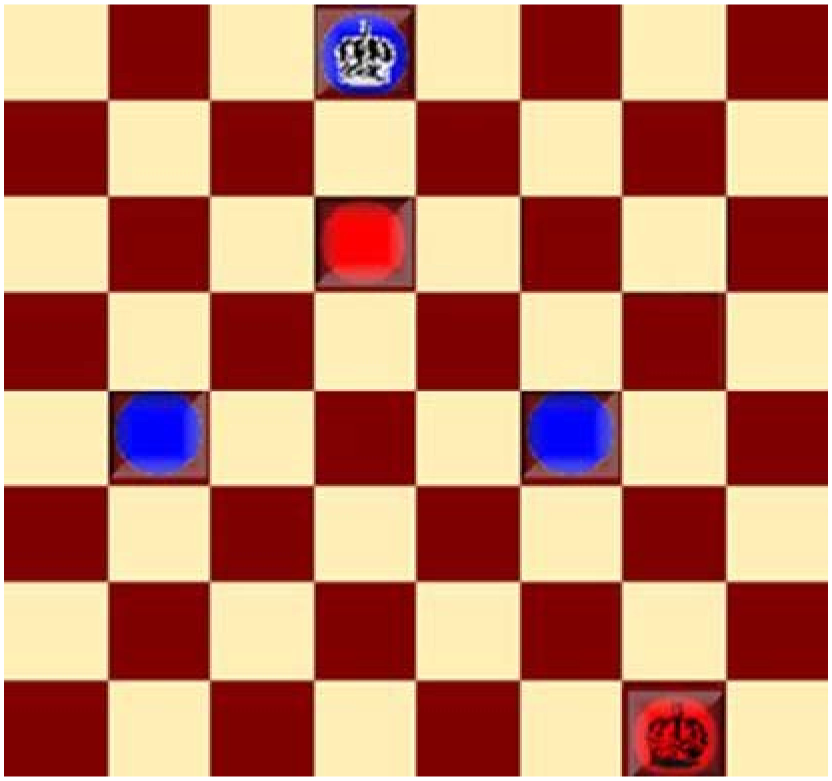
Objective: To gain programming experience in intelligent behavior planning in adversarial games.

In this assignment you will implement the Minimax algorithm for the game of checkers as described below.

- The initial board configuration contains 12 red single (**RS**) pieces and 12 blue single (**BS**) pieces:



- Moves are allowed only on the dark squares, so pieces always move diagonally. Single pieces are always limited to forward moves.
- A piece making a non-capturing move (not involving a jump) may move only one square.
- When a piece reaches the furthest row from the player who controls that piece, it is crowned and becomes a king. Note that kings are limited to moving diagonally but may move both forward and backward:



- A player wins the game when the opponent cannot make a move. In most cases, this is because all of the opponent's pieces have been captured, but it could also be because all of his pieces are blocked in.
- **MAX** always goes first and uses the following evaluation function:

  *E(s) = (5*BK + BS) - (5*RK + RS)*

  Where,

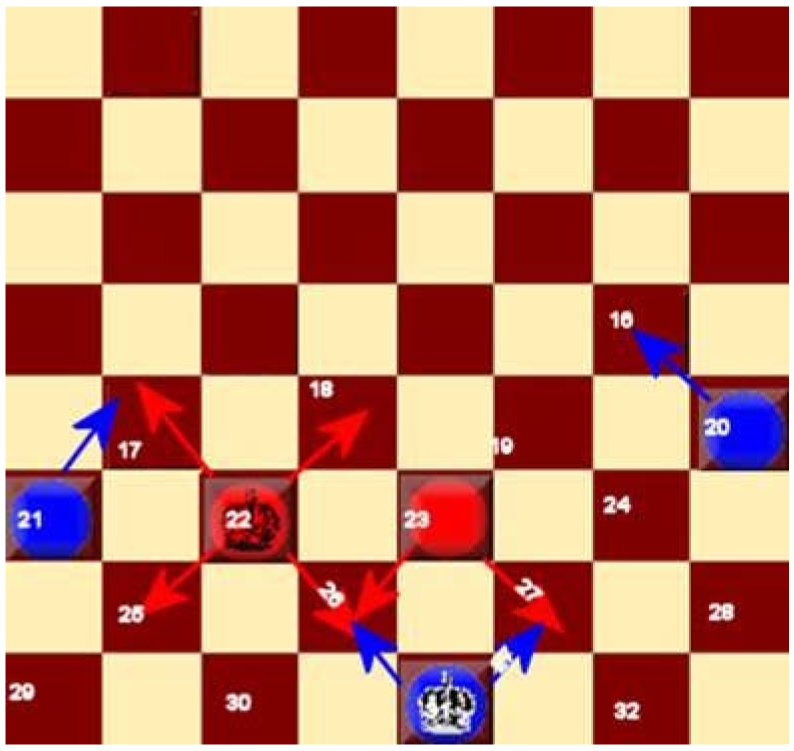
  BK = Blue king advantage

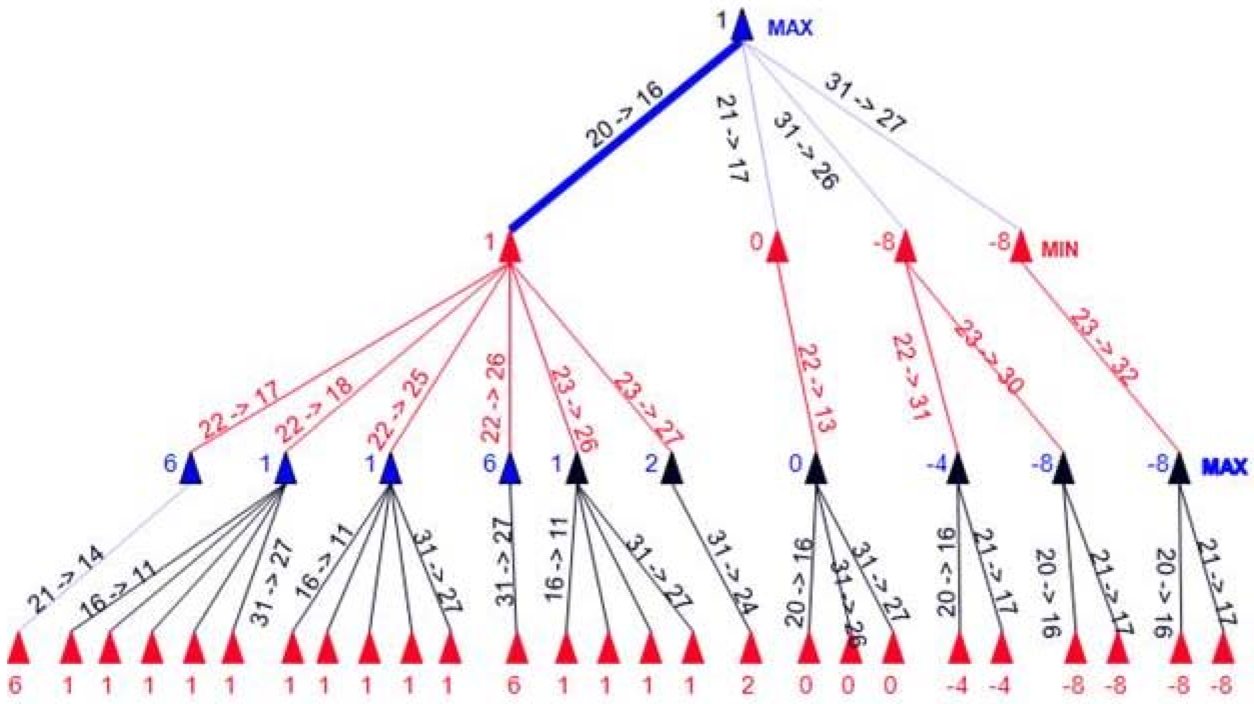
  BS = Blue single advantage

  RK = Red king advantage

  RS = Red single advantage

- Your project simulates the computer-versus-computer mode of adversarial game playing. So, have **MAX's** opponent use a less informed heuristic than the one shown above (e.g. the other player can be less aggressive in capturing its opponent's pieces).
- The game is 3-ply. For example, assuming we have the following pieces on the board on squares 16 through 32:





Programming Guidelines

- Your project simulates the computer-versus-computer mode of adversarial game playing.
- Your program should work with any initial configuration of pieces.
- The computer-computer game must continue until there is a winner.
- You must use my graphical front-end to implement and test your algorithm.

Input/Output Guidelines

1. To compile and run your application in the default mode:

```
javac Checkers.java
java Checkers
```

Grading Guidelines

- I do not accept multiple submissions
- Read my policy on late submissions

Technical Correctness • **Program compiles and executes correctly** • **Proper implementation of the Minimax algorithm** • **Assignment objectives are met** • **Specified I/O guidelines are followed**	**90%**
Coding Style & Documentation • **Opening comments (author name, assignment number, date, purpose, etc.) are included** • **General comments appear in code** • **Code is properly indented and spaced** • **Meaningful variable names are used** • **Code is properly modularized**	**10%**

Submission Guidelines

1. Place your program files (**Checkers.java**, etc.) and the supporting data files (if any) in a directory called **YourName-HW3** and archive it using Winzip or any other compression utility that you might have available on your PC. Include any special instructions to run your code in a **README** file.

2. Email me the zipped archive on or before the due date using the following subject line for your email message:

   Your Name – AI – HW3

   In addition, submit a **printout of all your program files** in class on the due date.