Rayhan Syed
Professor Sable
ECE-469

Project 1: Checkers

- The submission comes with a makefile to compile. Just need to type make to compile the program
- When running the program from the command line, there are the following optional flags:
    - -f: -f ExampleFile where ExampleFile is a text file that represents the board. From how the program is structured, checkers has 8 rows and 4 columns (as in every row, 4 of the squares are never occupied and thus can be in terms of data ignored). Thus the board cares for 32 positions, equating to a need for 32 characters. Each character can be of a few values, rather being an empty red/black square, a black or yellow piece, or a black or yellow king piece. Periods separate rows so 4-tuplets of characters compose a row. The file is parsed from bottom row to top mapping to rows 1 through 8. For example, an initial board would look like: 1111.1111.1111.----.----.2222.2222.2222.
        - Attached is a file named ExFile.txt that gives an example file where the front row for each side is composed of kings.
    - -n: -n # where # is a 2 bit binary number. A 0 represents a computer playing while a 1 represents a player. So -n 11 would be two human players paying in an alternating fashion while -n 00 is two computers playing. 10 is the default value for this number if no explicit -n flag is called.
- There are 2 prompts. The first asks which team should go first. The second sets a timer for how much time any computer player has on making decisions (this prompt while asked, does not apply when playing a human vs human game).
- Moving: All legal moves are listed by numbering from 1 to however many options are available. Entering the matching number for a given move will choose the move when pressing enter. If a number entered is not one of the allowed options, it is ignored and the user can choose another move. The format of options are given coordinates along the board [origin]->[dest]". If the piece is jumpin, a carat (^) character is used to indicate that. Board locations are described using [row.column] format. If there are multiple options in which the player can jump twice or more in a turn, the subsequent jump will be offered after choosing the initial jump.
- Certain requirements are needed for the computer to offer a draw, where the computer believes they are equal in position.
    - All pieces are kings
    - The players have equal # of pieces and the number is less than or equal to 3
    - No jumps are available to the next player.
- A user can offer a draw by typing "draw" at move-choice prompt. In a game against a computer, this is unnecessary, because if a computer is willing to accept a draw, it will offer it to you as well.
- I use multiple classes to implement checkers having 5 different sets of cpp and h files past main.cpp.

- Game: For a given player, the legal moves are enumerated. If its a human's turn, the player is prompted for a choice. At the end of each turn, the program checks to see if the conditions for the end of game has been met.
- Move : Moves in checkers can be seen as having a start/origin and end/destination. It is possible that one turn can be composed of multiple moves if jumps are taking place. In the case that there is a jump, that along with the origin tile is cleared. Since subsequent jumps are possible, a move is a vector with pointers that point to subsequent moves that would be the additional jumps.
- Board – This is how data is properly stored and organized regardless of player. IT obtains the legal moves for a given player, allows for what moves to be made, and accounts for how the board state changes. It also prints out the board to standard output. Moreover, it gets the best moves, uses mini-max search with alpha-beta pruning and contains the heuristic function for deciding the best move.
- The heuristic function weighs various values of the current board state to make decisions. These are as follows
    - How many more pieces one player has over another
    - How close any given piece that is not a king, is close to getting to the end of the board and being promoted
    - The general number of pieces on the board in order to measure how far the game has gone on. A winner promotes actions that minimize the number
    - Distance from opponent pieces as a winner needs to be near the opponent to capture pieces.
    - Other conditions related to the state of the board in certain areas such as the back sides and center and how full these areas are.